Real-Time Clock and Interrupt Module (RCIM) User's Guide

concurrent REAL-TIME

REDHAWK

0898007-1240
Jan 2025

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Real-Time 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 Real-Time, 800 NW 33rd Street, Pompano Beach, FL 33065. Mark the envelope **"Attention: Real-Time OS Publications Department**."

Printed in U. S. A.

Revision History:	Level:	Effective With:
Original Release -- August 2002	000	RedHawk Linux Release 1.1
Previous Release -- December 2003	210	RedHawk Linux Release 2.0
Current Release -- May 2005	300	RedHawk Linux Release 2.3
Update -- September 2005	310	RedHawk Linux Release 2.3-4.1
Update -- May 2007	320	RedHawk Linux Release 4.2
Update -- April 2008	330	RedHawk Linux Release 5.1
Update -- June 2008	400	RedHawk Linux Release 5.1
Update -- October 2010	500	RedHawk Linux Release 5.4
Update -- December 2011	600	RedHawk Linux Release 6.0
Update -- February 2013	610	RedHawk Linux Release 6.3
Update -- February 2014	620	RedHawk Linux Release 6.3
Update -- March 2016	700	RedHawk Linux Release 7.2
Update -- June 2016	800	RedHawk Linux Release 7.2
Update -- June 2017	900	RedHawk Linux Release 7.3
Update -- March 2021	1000	RedHawk Linux Release 8.2
Update -- June 2021	1100	RedHawk Linux Release 8.2
Update -- August 2021	1200	RedHawk Linux Release 8.2
Update -- April 2022	1210	RedHawk Linux Release 8.4
Update -- June 2023	1220	RedHawk Linux Release 9.2
Update -- May 2024	1230	RedHawk Linux Release 9.2

Preface

Scope of Manual

This manual is intended for users responsible for the installation and use of the Real-Time Clock and Interrupt Module (RCIM) on Concurrent Real-Time's iHawk™ systems under the RedHawk™ Linux® operating system.

NOTE

Two RCIM models are described in this guide: RCIM III and RCIM IV. The use of the term "RCIM" refers to functionality common to both boards. "RCIM III" and "RCIM IV" refer to the specific boards. Refer to the section "Specifications" on page 1-3 for specifications for each of the boards.

NOTE

Both the RCIM III and RCIM IV are PCI Express boards. For information covering the older RCIM I (PCI) and RCIM II (PCI-X) boards, refer to the Legacy RCIM User's Guide which can be found in Concurrent's Documentation Library at this location: https://redhawk.concurrent-rt.com/docs/

Structure of Manual

This manual consists of the following:

- Chapter 1, *Introduction*, contains a general overview and specifications for the RCIM boards.
- Chapter 2, *Hardware, Installation and Configuration*, provides a detailed description of the RCIM boards and connectors, as well as installation and configuration instructions.
- Chapter 3, *Functional Description*, provides the general operation, user interfaces and configuration options for the clocks and interrupts available on the RCIM.
- Chapter 4, *GPS Clock Synchronization*, covers the optional GPS module.
- Chapter 5, *IRIG Timecode Synchronization*, covers using the optional IRIG module.
- Appendix A, *RCIM IV Registers*, describes the RCIM IV registers.
- Appendix B, *RCIM III Registers*, describes the RCIM III registers.

- Appendix C, *Calculating RCIM Cable Propagation Delays,* provides a formula for guarding against propagation delay when chaining RCIMs.

Syntax Notation

The following notation is used throughout this guide:

italic	Books, reference cards, and items that the user must specify appear in *italic* type. Special terms may also appear in *italic*.
`list bold`	User input appears in **`list bold`** type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in **`list bold`** type.
`list`	Operating system and program output such as prompts, 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

Title	Pub No.
RedHawk Linux Release Notes Version x.x	0898003
RedHawk Linux User's Guide	0898004
RedHawk Linux Frequency-Based Scheduler (FBS) User's Guide	0898005

where *x.x* = release version

Contents

Chapter 4 GPS Clock Synchronization

Chapter 5 IRIG Timecode Synchronization

Appendix A RCIM IV Registers

Appendix B RCIM III Registers

Appendix C Calculating RCIM Cable Propagation Delays

1
Introduction

This chapter provides an overview and specifications for the Real-Time Clock and Interrupt Module (RCIM).

NOTE

Two RCIM models are described in this guide: RCIM III and RCIM IV. The use of the term “RCIM” refers to functionality common to both boards. “RCIM III” and “RCIM IV” refer to the specific boards. The section “Specifications” provides specifications for each of the boards.

NOTE

Both the RCIM III and RCIM IV are PCI Express boards. For information covering the older RCIM I (PCI) and RCIM II (PCI-X) boards, refer to the Legacy RCIM User’s Guide which can be found in Concurrent’s Documentation Library at this location: https://redhawk.concurrent-rt.com/docs/

Overview

The Real-Time Clock and Interrupt Module (RCIM) is a PCI-based card that supports time-critical applications that require rapid response to external events, synchronized clocks and/or synchronized interrupts.

When RCIM boards of various systems are chained together, an interrupt can be simultaneously distributed to all connected RCIMs, and from the RCIMs to all the associated host systems.

A synchronized high-resolution clock is provided so that all the RCIMs in an RCIM chain on multiple systems can share a common time base. It also provides a local POSIX 1003.1 compliant high resolution clock. An optional GPS module allows alignment of the clock to GPS standard time. Multiple RCIMs equipped with the GPS module can provide a common time base without cable connections. A high stability oscillator is standard. Optional oscillators improve the accuracy of times measured with the RCIM.

In addition to the clocks, this multi-purpose PCI-based card has the following functionality:

- connection of external device interrupts
- real-time clock timers that can interrupt the system

- programmable interrupt generators which allow generation of an interrupt from an application program

These functions can all generate local interrupts on the system where the RCIM card is installed. When systems are chained together, multiple input and output interrupts can be distributed to other RCIM-connected systems. This allows one timer or one external interrupt or one application program to interrupt multiple RedHawk Linux systems almost simultaneously to create synchronized actions.

Specifications

Feature	RCIM IV	RCIM III
Clocks		
POSIX		
Length	64 bits (two 32-bit words)	64 bits (two 32-bit words)
Resolution	High-order 32 bits–1 second Low-order 32 bits–400 nsec	High-order 32 bits–1 second Low-order 32 bits–400 nsec
Oscillator stability	+/-2.5 PPM	+/-2.5 PPM
Tick Timer		
Length	64 bits (two 32-bit words)	64 bits (two 32-bit words)
Resolution	64 bit counter of 400 ns ticks	64 bit counter of 400 ns ticks
Real-Time Clocks		
Number	8	8
Length	32 bits	32 bits
Resolution	1 microsecond (larger values programmable)	1 microsecond (larger values programmable)
Oscillator stability	+/-2.5 PPM	+/-2.5 PPM
Local Interrupts		
External Input Interrupts	12 shared inputs and/or outputs (3.5V or 5V TTL)	12 (5V TTL)
External Output Interrupts		12 (5V TTL)
Real-Time Clocks	8	8
Distributed Interrupts		
Input	12	12
Output	12	12
Interrupt Response Time		
Interrupt to user process	< 8 microseconds	< 8 microseconds
Packaging		
Form Factor	PCIe	PCIe
Maximum cable length (See Appendix C for calculations.)	30 meters	30 meters
External Connectors	26-pin HD D-SUB	Molex LFH-60
PCI Performance	x1	x1
Options	Multi-GNSS Timing Module, Oven Oscillator	GPS Module, Oven Oscillator
Environmental		
Operating Temperature	10^{o} to 55^{o} C	10^{o} to 55^{o} C
Storage Temperature	-40^{o} to 70^{o} C	-40^{o} to 70^{o} C
Relative Humidity	10 to 90% (non-condensing)	10 to 90% (non-condensing)
Power		
Consumption	~20 watts max	~20 watts max

2
Hardware, Installation and Configuration

This chapter provides a description of the RCIM PCI-based boards as well as installation and configuration information.

Board Descriptions

This section provides illustrations and descriptions of the RCIM IV and RCIM III boards.

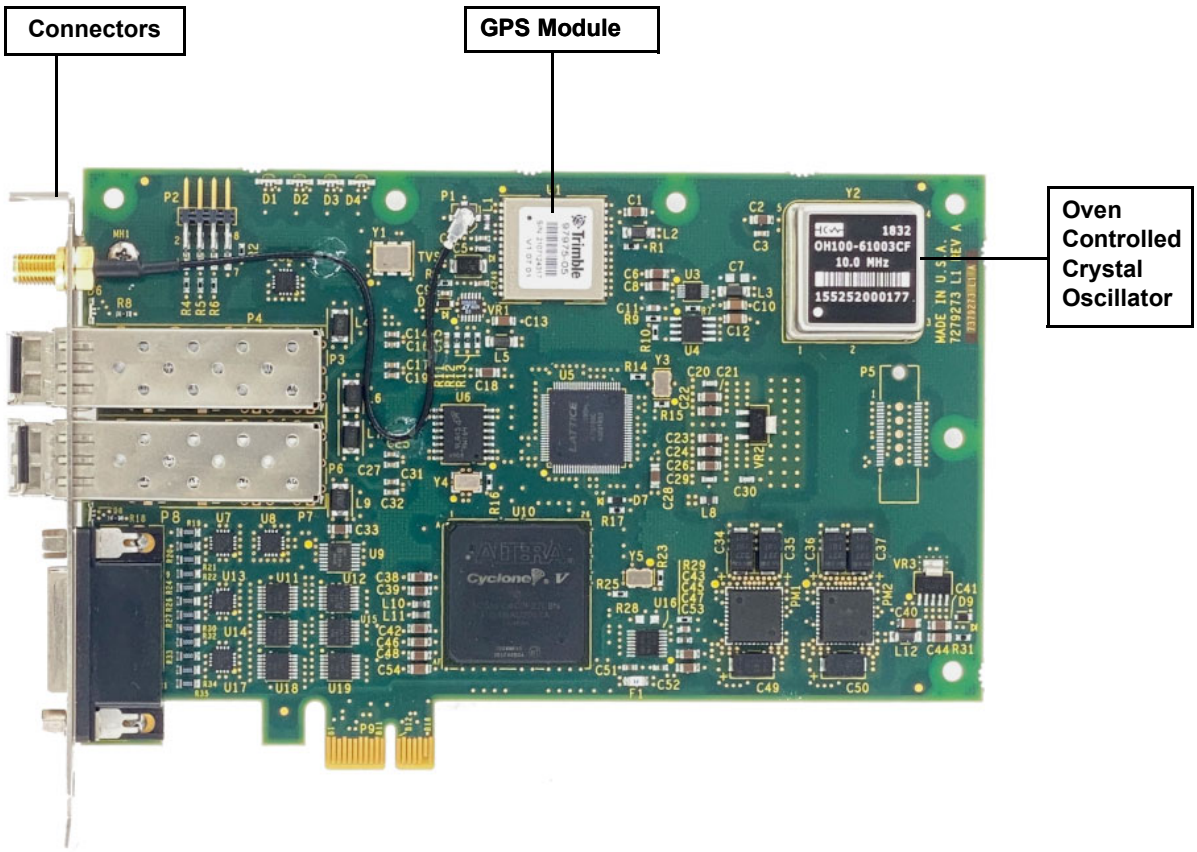
The RCIM boards mount into a standard PCI Express slot on a host system. A connector is mounted on each RCIM for connection to external interrupts, and a synchronization cable is included for daisy-chaining a master RCIM to one or more slave RCIMs.

RCIM IV

Board Illustration

Figure 2-1 shows the RCIM IV board with optional high stability OCXO (Oven Controlled Crystal Oscillator) and GPS modules installed.

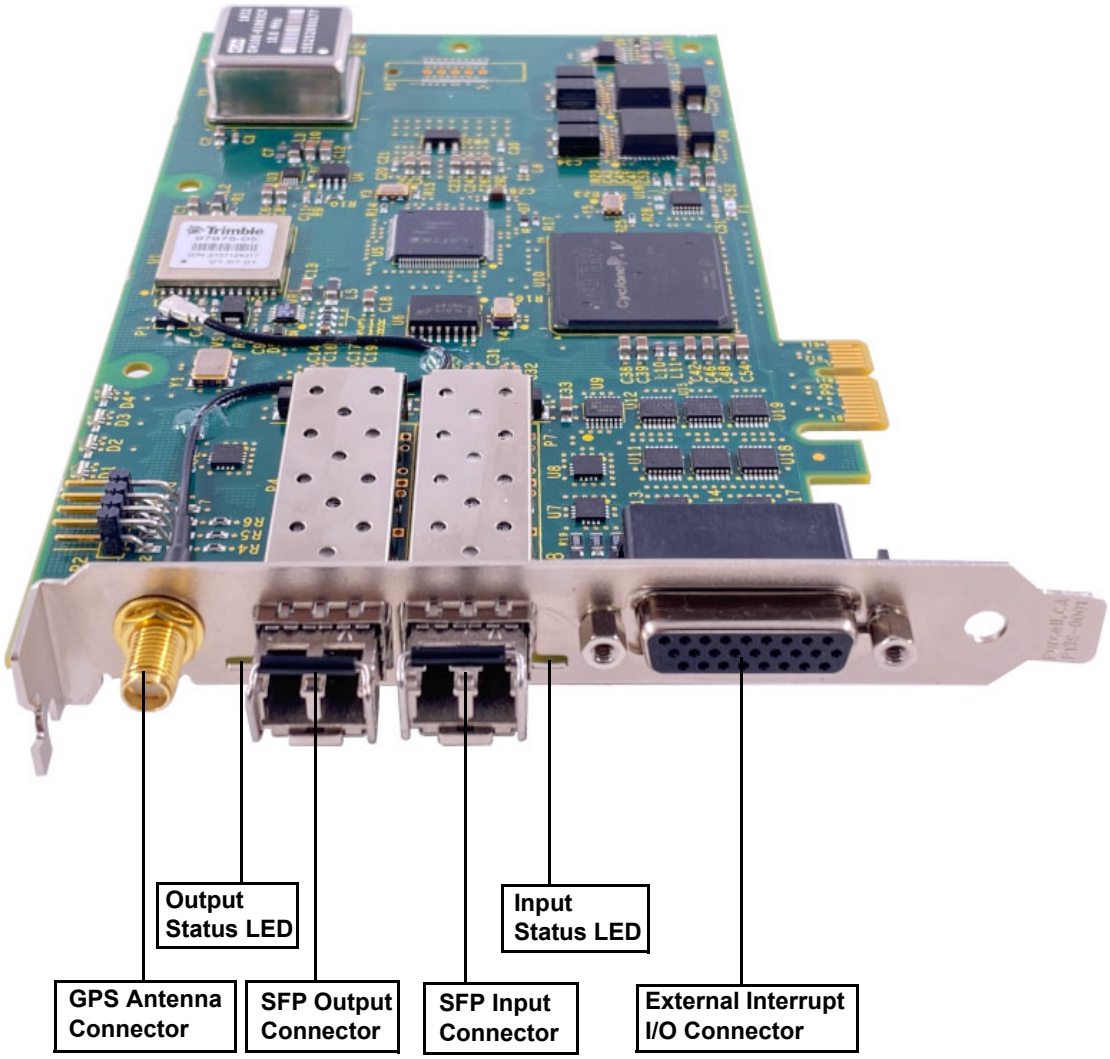
Figure 2-1 RCIM IV Board



Connectors and LEDs

Figure 2-2 shows the input/output connectors and LEDs on the RCIM IV board. Detailed information on the LEDs and each of the connectors is provided in the following sections.

Figure 2-2 RCIM IV Connectors and LED Locations



LED Functions

There are two bi-colored status LEDs near the input and output connectors on the RCIM IV board. They will both glow dimly RED when the board is in reset mode

followed by brief intervals of bright RED and GREEN as a test. During normal operation of the board the LEDs function as follows:

LED	description	Function
Output Status LED	RED solid	10 MHz clock failure
	RED 2/sec flash	POSIX clock stopped, with cable option, but cable not synchronized or missing
	GREEN 1/sec flash	POSIX clock running, without cable option
	GREEN 1/sec blink	POSIX clock running, with cable option, and cable synchronized
	RED/GREEN alternating 2/sec flash	POSIX clock running, with cable option, but cable not synchronized or missing
	BLUE 1/sec flash	POSIX clock running, without cable option, and receiving a valid GPS pulse
	BLUE 1/sec blink	POSIX clock running, with cable option, receiving a valid GPS pulse, and cable synchronized
	RED/BLUE alternating 2/sec flash	POSIX clock running, with cable option, receiving a valid GPS pulse, but cable not synchronized or missing
Input Status LED	RED 2/sec flash	with cable option, but cable not synchronized or missing
	GREEN solid	with cable option, and cable synchronized
	BLUE 1/sec flash	without cable option, but receiving a valid IRIG pulse
	BLUE 1/sec blink	with cable option, receiving a valid IRIG pulse, and cable synchronized
	RED/BLUE alternating 2/sec flash	with cable option, receiving a valid IRIG pulse, but cable not synchronized or missing

Input and Output Cables and Connectors

The RCIM IV uses a pair of standard SFP (small form-factor pluggable) connectors installed in cages to interface to the RCIM IV cable. The cable is used to communicate interrupts, time stamps and a reference clock between RCIM IV boards. The output cable connector is used when the RCIM is either the master or a slave in the middle of an RCIM chain (see page 2-14 for a description of RCIM modes). The input cable connector is used when the RCIM is acting in slave mode or in the middle of an RCIM chain. The cable part number (HS002-3CBL-xx where xx is length in meters) includes an LC fiber optic cable and two SFPs that are installed in the empty cages on the master and slave RCIMs. Refer to the section "Daisy Chain Cable" for more information about the cable.

NOTE

The cable SFPs should only be installed and removed with the system containing the RCIM IV powered down. See the Installation section for ESD caution. Care should be taken to insure that the SFP modules lock into position and that the RCIM IV is not pushed out of its PCIe slot during the installation of the SFPs. The fiber optic cables themselves can be installed and removed at any time without damaging the RCIM IV.

Oscillators

The temperature compensated crystal oscillator (TCXO) provided with RCIM IV has an accuracy of +/- 2.5 PPM (parts per million).

One optional oven controlled crystal oscillator (OCXO) provides a temperature stability of +/- 10 PPB (parts per billion).

GPS Antenna

The GPS option on the RCIM IV includes an active GPS antenna and coaxial cable.

The antenna receives the GPS satellite signals and passes them to the receiver. The GPS signals are spread spectrum signals in the 1575 MHz range and do not penetrate conductive or opaque surfaces. Therefore, the antenna must be located outdoors with a clear view of the sky.

If a different antenna or cable is used, it should match the following specifications:

- 50 Ohm impedence
- 27 dB gain
- 3.3 volt DC power max 30 ma.

External Interrupt I/O Connector

The external interrupt I/O connector on the RCIM IV is an industry standard 26-pin High Density D-SUB that provides twelve shared outputs and/or inputs.

By default, the RCIM IV external input and output signals are **high true** but this can be reversed. See “RCIM IV Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers” on page A-7 for more information.

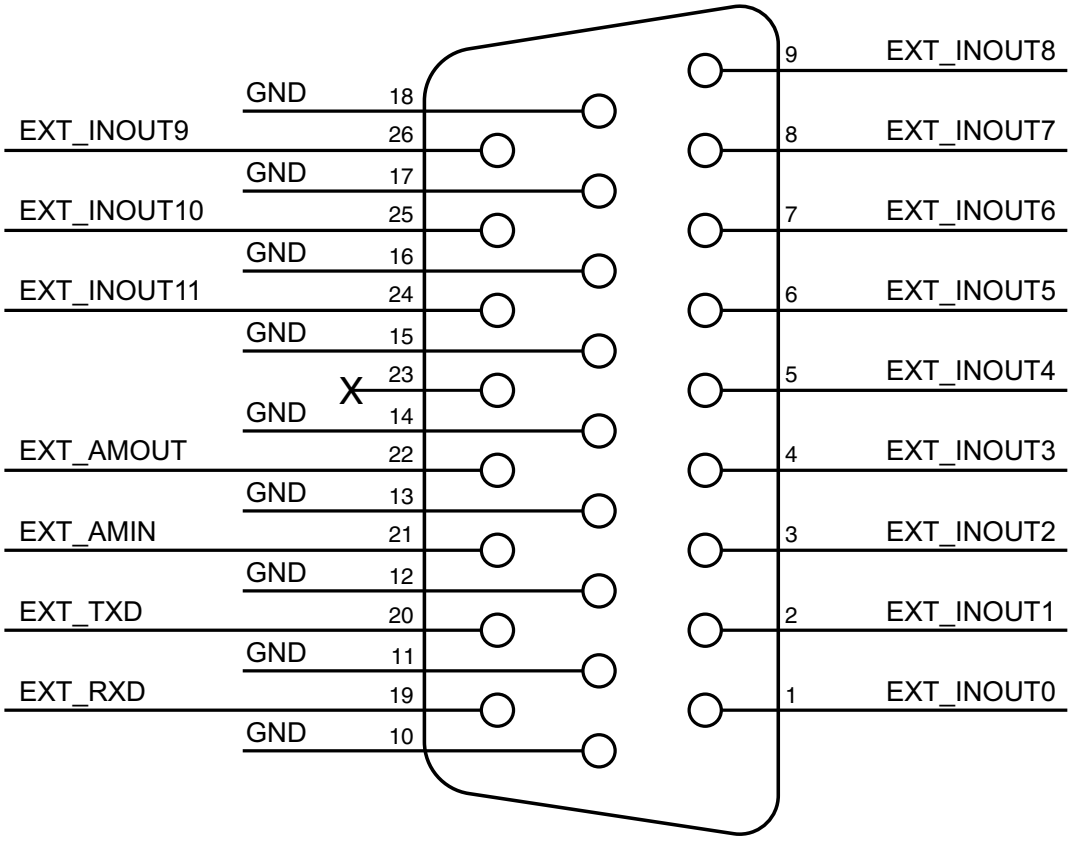
The external EXT_INOUT pins allow equipment to communicate with the RCIM via I/O interrupts that are software selectable to be either inputs or outputs with a switchable 100 Ohm termination. The function of each pin is driven by a multiplexer that can set individual pins to be either an input or an output. Pins set as outputs may be driven by any of the following RCIM devices: programmable interrupt generators (PIGs), real-time clock timers, (RTCs), edge-triggered interrupts (also known as input pins or ETIs), or distributed interrupts (DIs). Each pin is physically wired to be both an input and an output with the ability to disable the output gate. If no external wire is connected to an RCIM pin, the pin can act as both an input and output simultaneously for loopback connections.

By default, pins EXT_INOUT0 through EXT_INOUT5 are configured to be both inputs and outputs, and pins EXT_INOUT6 through EXT_INOUT11 are configured to be only inputs. See "RCIM IV Pin Configuration" on page 3-23 for more information.

See Chapter 3 for information on using external output interrupts and programmable interrupts.

The pin-outs for the external interrupt I/O connector are shown in Figure 2-3.

Figure 2-3 RCIM IV External Interrupt I/O Connector Pin-outs



RCIM IV external interrupt inputs have a selectable termination of 100 Ohms and are capable of 3.3V or 5V TTL levels. The recommended input signal duration is 1us and because termination can be disabled there is no minimum amperage requirement for line drivers.

The RCIM IV supports two programmable input sources, DCLS input and an external 10Mhz clock, that can be assigned to a physical pin to discipline the masterclock. The DCLS input will enable digital IRIG input, and the external 10Mhz clock input will automatically discipline the RCIM masterclock regardless of what other clocksources exist on the RCIM.

The signals EXT_RXD and EXT_TXD are RS-232 level signals. They are currently used for debug purposes.

System Identification

The following output to **`lspci(8)`** shows the PCI class, vendor and device IDs for the RCIM IV (1b:00.0 (*bus*:*slot*.*function*) will differ on your system):

```
# lspci -v | grep -i rcim
1b:00.0 System peripheral: Concurrent Real-Time RCIM-IV Real-Time
Clock & Interrupt Module (PCIe) (rev 01)
        Kernel driver in use: rcim
        Kernel modules: rcim

# lspci -ns 1b:00.0
1b:00.0 0880: 1542:9273 (rev 01)
```

Daisy Chain Cable

The RCIM IV uses a fiber optic serial synchronization cable with SFP (small form-factor pluggable) connectors (part no. HS002-3CBL-xx) to connect RCIM IVs in an RCIM chain. The serial data on the cable includes parity and framing information which allow cable problems to be detected. Polling is done continuously and messages that report the status of the RCIM IV daisy chain cables are output when an error condition is detected. Messages indicating problems will appear on the systems directly connected by a failing link.

The serial cables are point to point connections. The “input” cable refers to the cable going upstream towards the master RCIM. The “output” cable is the downstream connection away from the master.

```
RCIM: Input cable disconnected.
RCIM: Input cable connected.
RCIM: Input cable connected but not synchronized.
RCIM: Input cable unsynchronized.
RCIM: Input cable O.K.

RCIM: Output cable disconnected.
RCIM: Output cable connected.
RCIM: Output cable connected but not synchronized.
RCIM: Output cable unsynchronized.
RCIM: Output cable O.K.

RCIM: Cable error on input cable.
RCIM: Cable error on output cable.
```

The “disconnected” and “connected” messages will only occur based on whether an SFP is installed in the appropriate cage of the RCIM IV.

They will not occur when the optical cable is inserted or removed. They should not occur during normal operation unless the SFP is not installed correctly or it is malfunctioning.

The “not synchronized” and “unsynchronized” messages indicate that the cable is not answering attempts to communicate. These messages will occur when the optical cable is installed or removed. They will also occur when a connected system is powered off.

The last two messages indicate transient errors such as cable parity errors or temporary loss of cable synchronization. If a transient error occurs, it may require a link in the cable to resynchronize. If a distributed interrupt is being broadcast on the cable, it may be lost.

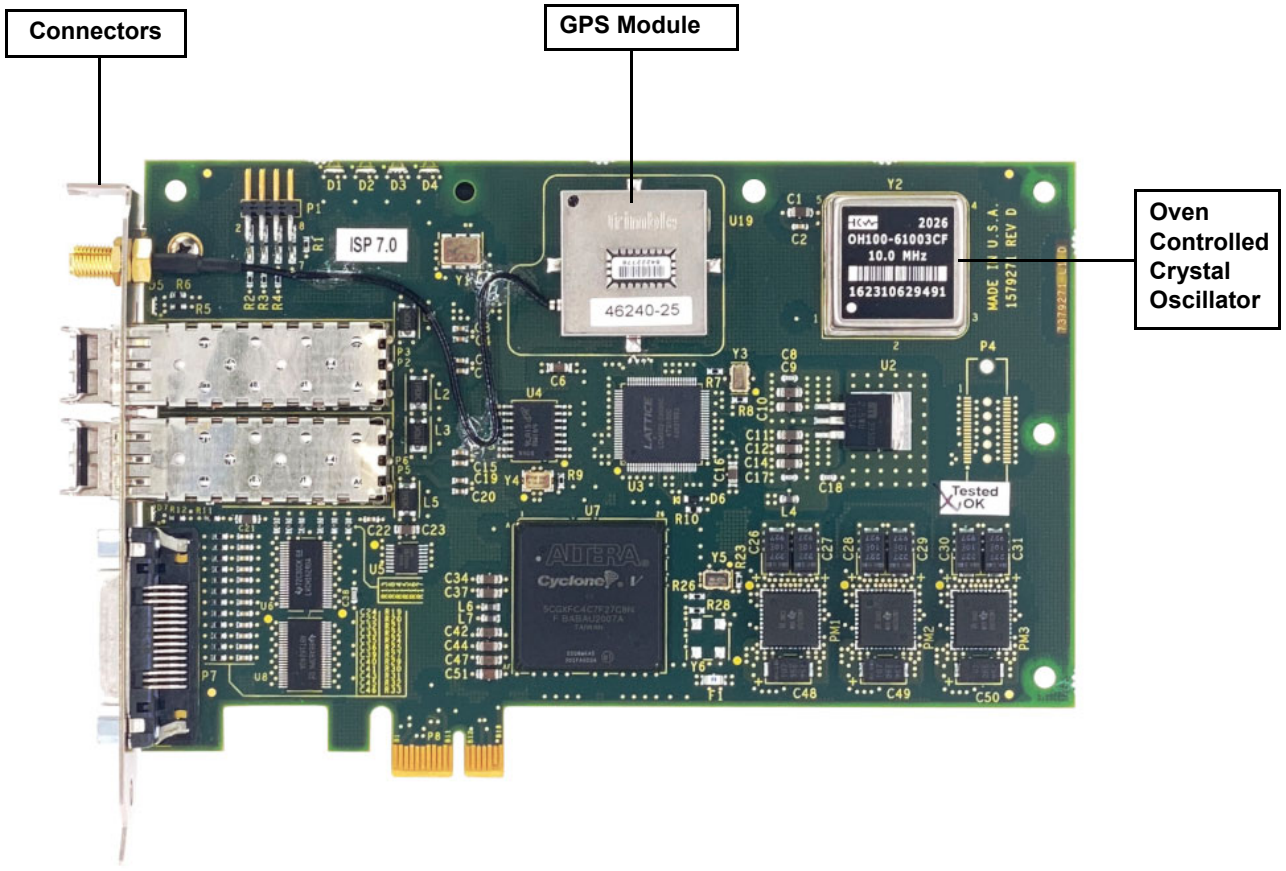
Transient errors also affect the synchronization of the tick timers since the cable clock will not reach all of the systems. Refer to Chapter 3 for instructions for synchronizing clocks.

RCIM III

Board Illustration

Figure 2-4 shows the RCIM III board with optional high stability OCXO (Oven Controlled Crystal Oscillator) and GPS modules installed.

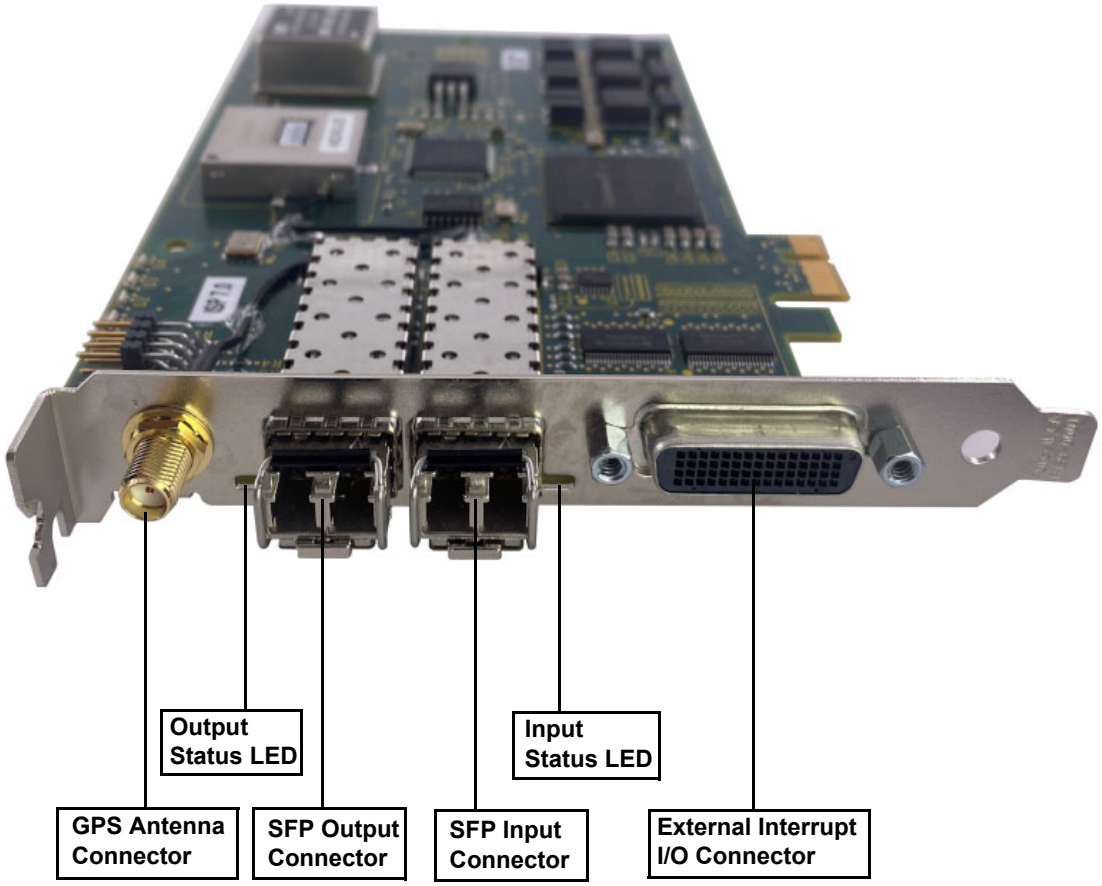
Figure 2-4 RCIM III Board



Connectors and LEDs

Figure 2-5 shows the input/output connectors and LEDs on the RCIM III board. Detailed information on the LEDs and each of the connectors is provided in the following sections.

Figure 2-5 RCIM III Connectors and LED Locations



LED Functions

There are two bi-colored status LEDs near the input and output connectors on the RCIM III board. They will both glow dimly RED when the board is in reset mode

followed by brief intervals of bright RED and GREEN as a test. During normal operation of the board the LEDs function as follows:

LED	description	Function
Output Status LED	RED solid	10 MHz clock failure
	RED 2/sec flash	POSIX clock stopped, with cable option, but cable not synchronized or missing
	GREEN 1/sec flash	POSIX clock running, without cable option
	GREEN 1/sec blink	POSIX clock running, with cable option, and cable synchronized
	RED/GREEN alternating 2/sec flash	POSIX clock running, with cable option, but cable not synchronized or missing
Input Status LED	RED 2/sec flash	with cable option, but cable not synchronized or missing
	GREEN solid	with cable option, and cable synchronized

Input and Output Cables and Connectors

The RCIM III uses a pair of standard SFP (small form-factor pluggable) connectors installed in cages to interface to the RCIM III cable. The cable is used to communicate interrupts, time stamps and a reference clock between RCIM III boards. The output cable connector is used when the RCIM is either the master or a slave in the middle of an RCIM chain (see page 2-14 for a description of RCIM modes). The input cable connector is used when the RCIM is acting in slave mode or in the middle of an RCIM chain. The cable part number (HS002-3CBL-xx where xx is length in meters) includes an LC fiber optic cable and two SFPs that are installed in the empty cages on the master and slave RCIMs. Refer to the section "Daisy Chain Cable" for more information about the cable.

NOTE

The cable SFPs should only be installed and removed with the system containing the RCIM III powered down. See the Installation section for ESD caution. Care should be taken to insure that the SFP modules lock into position and that the RCIM III is not pushed out of its PCIe slot during the installation of the SFPs. The fiber optic cables themselves can be installed and removed at any time without damaging the RCIM III.

Oscillators

The temperature compensated crystal oscillator (TCXO) provided with RCIM III has an accuracy of +/- 2.5 PPM (parts per million).

One optional oven controlled crystal oscillators (OCXO) provides a temperature stability of 10 PPB (parts per billion).

GPS Antenna

The GPS option on the RCIM III includes an active GPS antenna and coaxial cable.

The antenna receives the GPS satellite signals and passes them to the receiver. The GPS signals are spread spectrum signals in the 1575 MHz range and do not penetrate conductive or opaque surfaces. Therefore, the antenna must be located outdoors with a clear view of the sky.

If a different antenna or cable is used, it should match the following specifications:

- 50 Ohm impedence
- 27 dB gain
- 3.3 volt DC power max 30 ma.

External Interrupt I/O Connector

The external interrupt I/O connector on the RCIM III is a Molex LFH-60 (Low Force Helix) that provides twelve outputs and twelve inputs.

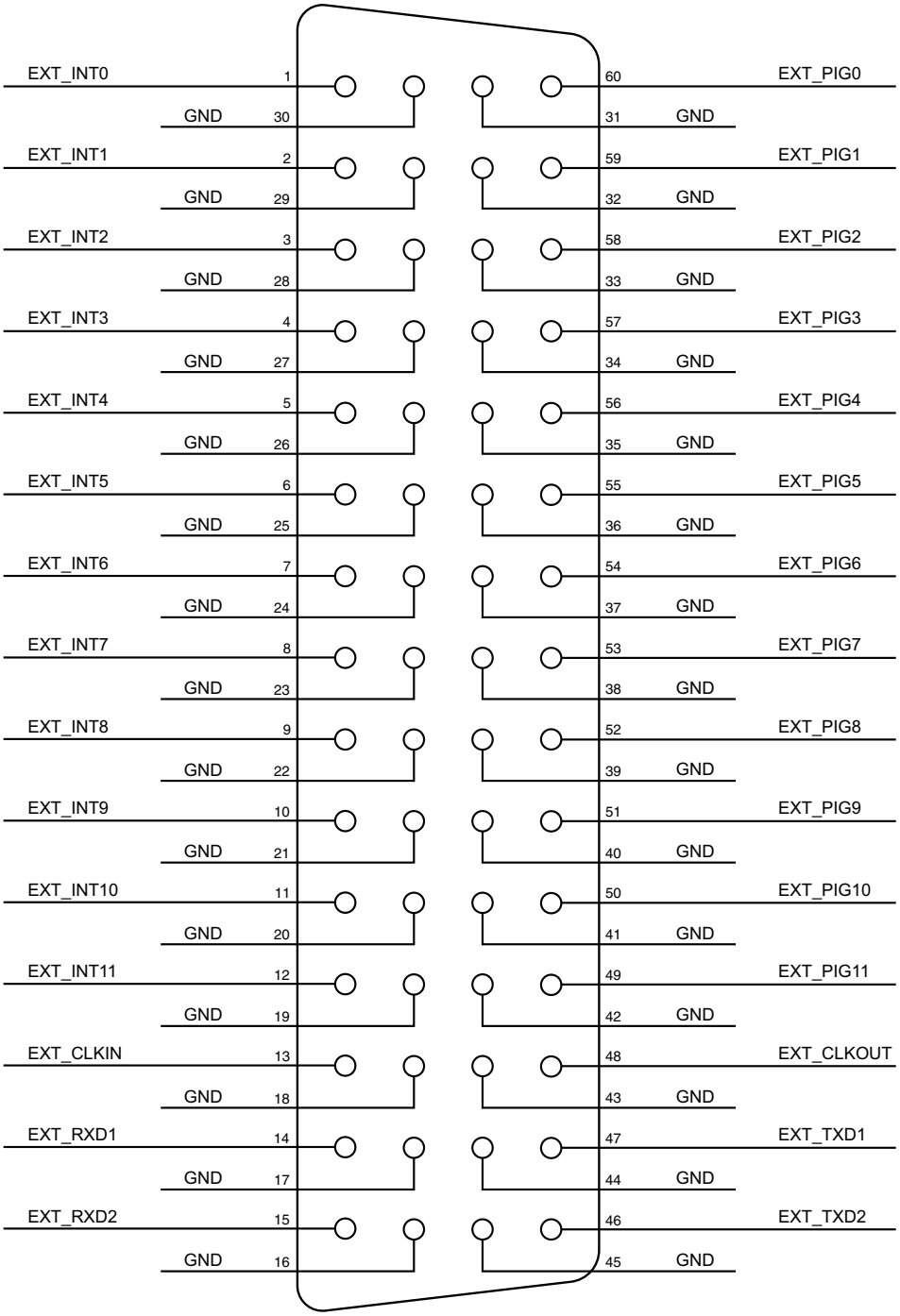
By default, the RCIM III external input and output signals are **low true** but this can be reversed. See “RCIM III Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers” on page B-6 for more information.

The external outputs allow equipment to be attached and controlled by the RCIM. The outputs are driven by a multiplexer which can select any of the programmable interrupt generators (PIGs), real-time clock timers (RTCs), edge-triggered interrupts (ETIs) or distributed interrupts (DIs) to drive the output. The selection is controlled by a set of configuration registers.

See Chapter 3 for information on using external output interrupts and programmable interrupts.

The pin-outs for the external interrupt I/O connector are shown in Figure 2-6.

Figure 2-6 RCIM III External Interrupt I/O Connector Pin-outs



The external interrupt input signals are 5 volt TTL levels. The external interrupt outputs (labeled EXT_PIG[0-11]) are driven using a 74ABT16240 line driver. The external interrupt inputs are terminated with 180 ohms to +5 volts, 330 ohms and 0.1 uf to ground. To drive this input requires a line driver that can sink at least 30 ma. The input termination limits the speed of the external interrupt signals and helps prevent noise from causing spurious interrupts. Since most line drivers can sink more current than they can source, the falling edge of the signal will be faster.

The signals EXT_CLKIN and EXT_CLKOUT are used for external 10MHz clocks in or out. An external clock driving the RCIM III should be capable of driving a 5V TTL signal into a 50 ohm load. The RCIM III will automatically switch to using the external clock if one is present. The external clock output from the RCIM III is driven using a 74ABT16240 line driver.

The signals EXT_RXD1, EXT_TXD1, EXT_RXD2, and EXT_TXD2 are RS-232 level signals. They are currently used for debug purposes.

System Identification

The following output to **`lspci(8)`** shows the PCI class, vendor and device IDs for the RCIM III (0e:04.0 (*bus*:*slot*.*function*) will differ on your system):

```
# lspci -v  | grep -i rcim
0e:04.0 System peripheral: Concurrent Real-Time, Inc. RCIM III Real-
Time Clock & Interrupt Module (PCIe) (rev 01)
# lspci -ns 0e:04.0
0e:04.0 Class 0880: 1542:9271 (rev 01)
```

Daisy Chain Cable

The RCIM III uses a fiber optic serial synchronization cable with SFP (small form-factor pluggable) connectors (part no. HS002-3CBL-xx) to connect RCIM IIIs in an RCIM chain. The serial data on the cable includes parity and framing information which allow cable problems to be detected. Polling is done continuously and messages that report the status of the RCIM III daisy chain cables are output when an error condition is detected. Messages indicating problems will appear on the systems directly connected by a failing link.

The serial cables are point to point connections. The “input” cable refers to the cable going upstream towards the master RCIM. The “output” cable is the downstream connection away from the master.

```
RCIM: Input cable disconnected.
RCIM: Input cable connected.
RCIM: Input cable connected but not synchronized.
RCIM: Input cable unsynchronized.
RCIM: Input cable O.K.

RCIM: Output cable disconnected.
RCIM: Output cable connected.
RCIM: Output cable connected but not synchronized.
RCIM: Output cable unsynchronized.
RCIM: Output cable O.K.

RCIM: Cable error on input cable.
RCIM: Cable error on output cable.
```

The “disconnected” and “connected” messages will only occur based on whether an SFP is installed in the appropriate cage of the RCIM III.

They will not occur when the optical cable is inserted or removed. They should not occur during normal operation unless the SFP is not installed correctly or it is malfunctioning.

The "not synchronized" and "unsynchronized" messages indicate that the cable is not answering attempts to communicate. These messages will occur when the optical cable is installed or removed. They will also occur when a connected system is powered off.

The last two messages indicate transient errors such as cable parity errors or temporary loss of cable synchronization. If a transient error occurs, it may require a link in the cable to resynchronize. If a distributed interrupt is being broadcast on the cable, it may be lost.

Transient errors also affect the synchronization of the tick timers since the cable clock will not reach all of the systems. Refer to Chapter 3 for instructions for synchronizing clocks.

Connection Modes

When RCIM boards of various systems are chained together, an interrupt can be simultaneously distributed to all connected RCIMs, and from the RCIMs to all the associated host systems.

NOTE

RCIMs in a chain can all be the same model, or they can be a mix of RCIM III and RCIM IV boards.

If your system will be part of an RCIM chain, it is best to determine the desired connection mode before installing an RCIM; it is easier to connect the optical cable to the cable connectors before the RCIM is installed.

Note that to reconfigure an RCIM chain, the systems must be powered off and rebooted after moving the cables. The driver determines if a system is the master RCIM at boot time and configures the master system to control the cable clock and its enable. Swapping the cables without rebooting the systems will result in problems with the cable clock.

An RCIM can be connected in one of the following modes:

Isolated mode	There are no connections to any other RCIM.
Master mode	The RCIM is at the head of a chain of RCIMs. There is no cable connection going into this RCIM, only a cable connection going out. The RCIM master is unique in that it controls the clocks (see Chapter 3 for a discussion).
Pass-through Slave mode	The RCIM is connected to two other RCIMs. There is an input cable connection coming from the previous RCIM in the chain, and an output cable connection going to the next RCIM in the chain.
Final Slave mode	The RCIM is connected to one other RCIM. There is an input cable connection going into a final slave RCIM but no output cable connection coming out of it.

Unpacking the RCIM

When unpacking the equipment from the shipping container, refer to the packing list and verify that all items are present. Save the packing material for storing and reshipping the equipment.

NOTE

If the shipping container is damaged upon receipt, request that the carrier's agent be present during unpacking and inspection of the equipment.

Installation

Normally, installation and configuration of the card is done by Concurrent Real-Time. This information is provided for those cases where an RCIM is added to a system in a post-manufacturing environment.

In order to successfully install the RCIM, you must know if you will be using the RCIM to accept or deliver external interrupts and the mode in which the RCIM will run (isolated, master, pass-through slave or final slave). Refer to the section "Connection Modes" above for details.

CAUTION

Avoid touching areas of integrated circuitry as static discharge can damage circuits.

Concurrent Real-Time strongly recommends that you use an anti-static wrist strap and a conductive foam pad when installing or upgrading a system. Electronic components such as disk drives, computer boards, and memory modules can be extremely sensitive to Electrostatic Discharge (ESD). After removing the board from the system or its protective wrapper, place it flat on a grounded, static-free surface, component side up. Do not slide the board over any surface.

If an ESD station is not available, you can avoid damage resulting from ESD by wearing an antistatic strap (available at electronic stores) that is attached to an unpainted metal part of the system chassis.

Use the following procedure to install an RCIM in your system:

1. Ensure that your system is powered down.
2. Remove the power cable from the system.
3. Open the case of your system and identify the PCIe slot where you want the RCIM to reside. In general, it is best for the RCIM to be configured in a slot where minimal or no contention with other devices occurs and at the highest IRQ priority possible.

4. Install the RCIM into the desired slot, securing the card in the slot using the mechanism provided by the case.
5. If this is to be part of an RCIM chain, attach the cable as required. See the section "Connection Modes" to determine how to connect the cable based on the connection mode for this system.
6. If you are installing an RCIM board equipped with the optional GPS module, attach the GPS antenna lead and mount the antenna. The antenna should be mounted on the rooftop or in an open area. Refer to the sections "Connectors and LEDs" and "GPS Antenna" more information about the antenna.
7. Replace the cover.
8. Attach the power cable to the system.
9. Apply power and boot the system.

Configuration

Kernel Configuration

The following RedHawk Linux kernel parameters are associated with the RCIM. All are accessible through the Character Devices selection of the Kernel Configuration GUI and are enabled by default in all the pre-built RedHawk Linux kernels.

RCIM — This parameter configures the RCIM driver in the kernel. It can be configured as a module, if desired.

RCIM_MASTERCLOCK — This parameter enables the RCIM to be used as a master clock that monitors and adjusts system time, resulting in more precise system timekeeping.

RCIM_IRQ_EXTENSIONS — This parameter allows other drivers to attach their own interrupt routines to the RCIM driver. The Frequency-based Scheduler (FBS) requires this support.

MULTI_RCIM_MAX — Default 24. This numeric parameter defines the maximum number of individual RCIM cards that can be configured into the system.

For complete information about modifying kernel tunables and building a kernel, refer to the *RedHawk Linux User's Guide*, publication number 0898004.

Driver Configuration

RCIM boards have many features that can be configured for application use. Configuration is performed by echoing a series of comma-separated tokens to the configuration file associated with the RCIM board of interest. Each token specifies how a single feature configuration option is to be changed; feature tokens not specified remain unchanged.

For example, the following command changes `eti1` edge-triggered interrupt to trigger on a falling edge:

```
echo "eti1/f" > /proc/driver/rcim:0/config
```

Quotes around the tokens are always recommended and must be used to surround a configuration request that contains a vertical bar, as in the following command example:

```
echo "rtc0|di1" > /proc/driver/rcim:0/config
```

After execution of the above command, the RCIM's `rtc0` real-time clock will be routed into the RCIM's `di1` distributed interrupt.

Here is a more complex example for an RCIM slave system:

```
echo "host/server1.ccur.com, eti1/rising, di3/high, rtc3|di6" >
/proc/driver/rcim:0/config
```

This command performs all of the following actions:

- sets the master RCIM hostname to `server1.ccur.com`
- sets the `eti1` edge-triggered interrupt to trigger on a rising edge
- sets the `di3` distributed interrupt to trigger on a high value
- routes the `rtc3` real-time clock into the `di6` distributed interrupt.

Note that all RCIM configuration options fall into one of the following catagories:

- Set the manner in which various interrupts will be triggered: rising or falling edge, high or low level.
- Create associations between internally generated interrupt signals, output lines and distributed-interrupt lines.
- Set the name of the system in an RCIM chain that has the master RCIM.
- Decide whether the tick and POSIX clock is to be driven by the local RCIM oscillator or by the master RCIM oscillator, for RCIMs that are in an RCIM chain.

Configuration changes require write permissions to the RCIM's config file (e.g. root access) and should only be made when the RCIM is not otherwise in use.

Configuration modifications are not automatically retained between system reboots. To make them persistant, modify the **`/etc/sysconfig/rcim`** file to create the desired RCIM configuration during each system boot.

Refer to the *Available Configuration Options* section of the **`rcim(4)`** man page for complete details on RCIM configuration.

General Considerations

When configuring the RCIM systems, keep the following in mind:

- For a chain of RCIMs, the tick clock and POSIX clock in all slave RCIMs will be synchronized with the master because the clock signal incrementing time in the master is broadcast to all slaves. Once the clocks on all RCIMs are initially synchronized they will remain synchronized.

  To synchronize tick clocks, a working TCP/IP connection between all systems is needed. In addition, each slave RCIM hostname configuration must be set to the master RCIM, and each slave must be configured to run the **rcim_clocksync** init script once on boot. This is only required if your application is using the tick clock for synchronization.

  System timekeeping daemons can be used to synchronize the POSIX clocks, however there is a better mechanism: **rcimdate**(8). The RCIM-master broadcasts its POSIX time down the RCIM cabling once per second; **rcimdate** uses this to make the slave POSIX clocks exactly match the master. This has several advantages over system timekeeping daemons because no TCP/IP connections between systems are required, synchronization is faster, and synchronization is extremely accurate.

- Interrupts, whether operating locally or distributed across an RCIM chain, will be processed according to the values configured on each system. If you wish them to function in a manner different from established defaults, the desired configuration options must be specified.

- When distributing interrupts across the systems in an RCIM chain, all systems must have a compatible configuration for the distributed interrupt lines.

- By default, RCIM-IV external input and output signals are **high true**, but RCIM-III external input and output signals are **low true**. Control registers exist on both to reverse the polarity if desired. For example, external interrupt input signals have polarity control for level interrupts and falling/rising edge control for edge interrupts, and programmable interrupt generator (PIG) outputs can be set or reset to any desired state.

MSI Interrupt Configuration

RCIM IV and the latest versions of RCIM III (revision 9 and later) support MSI (message signaled interrupts). By default, the RCIM kernel driver will initialize the hardware to use MSI interrupts instead of PCI INTA interrupts whenever possible. By using MSI interrupts, the RCIM is guaranteed of having its own non-shared interrupt, thus providing more reliable interrupt response times.

In addition, the RCIM IV fully supports the MSI-X standard and allows attaching individual sources (e.g., `rtc3`) to up to 16 different interrupts (e.g., `irq8`) on compatible hardware. See "MSI-X Interrupt Configuration" on page 3-17 for more information.

The RCIM driver has an `rcim.nomsi=1` option which is independent of the `rcim=` configuration option described earlier. All versions of the driver have this option. If specified the MSI capability is disabled on all RCIM boards that support it. There is no mechanism to pick which boards are to be MSI-disabled and which ones are not. When this option is specified, the RCIM driver will fallback to using the PCI INTA interrupt

method. For performance reasons, this option should only be used if a problem with MSI interrupts is encountered.

For a statically linked RCIM driver, this tunable can be specified on the GRUB boot loader command line (`rcim.nomsi=1`). For an RCIM in module form, this tunable can be placed in **`/etc/modprobe.conf`** as "`options rcim.nomsi=1`".

3
Functional Description

This chapter describes the clocks and interrupt capabilities provided by the RCIM and the user interfaces for each.

Overview

The Real-Time Clock and Interrupt Module (RCIM) provides two non-interrupting clocks. One of these clocks can be synchronized with all the RCIMs in an RCIM chain to provide a common time stamp across systems. The other clock is POSIX 1003.1 compliant and, although not synchronized across the RCIM chain, it increments in unison with the other clock on the RCIM board and can be set to a specific time.

In addition to the clocks, the following methods for handling signal processing (interrupts) are available:

- Edge-Triggered Interrupts (ETIs)
- Real-Time Clocks (RTCs)
- External Output Interrupts
- Programmable Interrupt Generators (PIGs)
- Distributed Interrupts (DIs)

These interrupts operate locally on an RCIM system or can be distributed across all RCIM systems in an RCIM chain. The **`open(2)`**, **`close(2)`** and **`ioctl(2)`** system calls are used to manipulate the interrupts. Separate device files are associated with each interrupt.

The clocks and interrupts are described in this chapter.

Clocks

The RCIM provides two non-interrupting clocks, which are fully explained in the sections that follow.

tick — a 64-bit non-interrupting clock that increments by one on each tick of the common 400ns clock signal. This clock can be reset to zero and synchronized across the RCIM chain, providing a common time stamp.

POSIX — a 64-bit non-interrupting clock encoded in POSIX 1003.1 format. The upper 32 bits contain seconds and the lower 32 bits contain nanoseconds. This clock is incremented on each tick of the common clock signal. It is primarily used as a high-resolution local clock. It can be configured to synchronize system time with GPS standard time on boards equipped with GPS.

All clocks on all RCIMs in the chain are incremented in unison, as they are all driven by a common clock signal emanating from the master RCIM.

The Tick Clock

The tick clock is a 64-bit non-interrupting counter that increments by one on each tick of the common clock signal. Although it cannot be set to a specific time, it can be incremented or set to zero. Hence the tick clock cannot be adjusted on the fly to approximate the current time of day as would be required of a true time-of-day clock.

When an RCIM board is part of an RCIM chain, the tick clocks on all slave RCIMs are incremented and cleared in synchronization with whatever incrementing and clearing is done to the tick clock located on the master RCIM.

The tick clock can be read on any system, master or slave, using direct reads when the device file **/dev/rcim:***N***/sclk** (where *N* is the RCIM card number starting from zero) is mapped into the address space of a program. See the section "Direct Access to the Clocks" below for details.

By default, tick clock initialization (zeroing) and synchronization with other tick clocks on the RCIM chain occur automatically whenever the RCIM master boots. Initializing and synchronizing tick clocks is accomplished using the **rcim_clocksync(1)** command. See the sections "Synchronizing the Tick Clock" and "The rcim_clocksync Utility" below for details.

The POSIX Clock

The POSIX clock is a 64-bit non-interrupting counter encoded in POSIX 1003.1 format. The upper 32 bits contain seconds and the lower 32 bits contain nanoseconds. This clock is incremented on each tick of the common clock signal and the system clock is synchronized to the POSIX clock.

Setting the system clock (for example, with **clock_settime**(2)) will set both the system clock and the RCIM POSIX clock to the new time. In addition, the POSIX clock can be mapped with `mmap` and then read like any other RCIM register by applications. However, modifying the RCIM POSIX clock via the `mmap` method is not recommended while the system is synchronizing with the RCIM POSIX clock.

The POSIX clock can be loaded with any desired time; however, the value loaded is not synchronized with other clocks in an RCIM chain. Only the POSIX clock of the RCIM attached to the host is updated. See the section "Direct Access to the Clocks" below for details about accessing the POSIX clock.

The **rcimdate** command can be run on each slave to synchronize the POSIX clock to that of the master. No TCP/IP connection or other software is needed. **rcimdate** utilizes the POSIX timestamp sent down the RCIM cabling once per second by the RCIM maser.

On an RCIM system with GPS module and system timekeeping software running (e.g. **ntpd** alone or **gpsd** with **chronyd**), the GPS receiver is used to synchronize the POSIX clock on the RCIM on which it is attached to GPS time. One GPS-equipped RCIM can synchronize all iHawks in an RCIM chain. Multiple RCIMs equipped with the GPS module can provide a common time base without any cable connections between systems. POSIX timers based on absolute GPS time can be used to simultaneously start the execution of programs on systems which are not physically connected.

See the section “Using GPS for System Timekeeping” for details. Generally, only the master RCIM needs a GPS; slaves use one of the software methods available for synchronizing their POSIX clock to the master.

Direct Access to the Clocks

The device file **/dev/rcim:***N***/sclk** (where *N* is the RCIM card number starting from zero) can be used to access the RCIM clocks directly using **mmap(2)**. From the address returned by **mmap**, the following offsets are used to access the clock fields:

0x0	upper 32-bits of tick clock
0x8	lower 32-bits of tick clock
0x10	status and control (cannot be modified)
0x100	POSIX clock seconds
0x108	POSIX clock nanoseconds
0x110	status and control (cannot be modified)

These offsets are defined in the header file **/usr/include/linux/rcim.h** (with names beginning with RCIM_SYNCCLOCK_).

To set the value of the POSIX clock, the **rcim_clocksync(1)** utility can be used in interactive mode using the “update” command.

Synchronizing the Clocks

The sections below describe the techniques and tools used to synchronize the clocks on the RCIM.

The rcim_clocksync Utility

The **rcim_clocksync(1)** utility can be used to reset the tick clocks on all connected RCIMs to zero to provide a common time stamp across the system. This synchronization operation occurs automatically when the RCIM master system boots. As slave systems become available, it will be necessary to reissue this command to synchronize the tick clocks across the RCIM chain, however, this can be automated (see the section “Automatic Synchronization” below). **rcim_clocksync** can also be used to synchronize the POSIX clocks on all connected RCIMs. This procedure is described in the section “Synchronizing the POSIX Clock.”

Note that the system clock is synchronized with the RCIM and when **rcim_clocksync** is run other than at system boot, there are consequences and should be used with caution. Master and slave system times are synchronized with the POSIX clock, and when system time stops advancing, time-based functions using those clocks will stop.

Synchronization always succeeds on the RCIM master or isolated system; an error is returned if synchronization is attempted on an RCIM slave system.

Specifying **rcim_clocksync** with no options on the RCIM master system synchronizes all tick clocks in the RCIM chain.

`rcim_clocksync` takes the following options:

-i interactive mode (see below)

-m prints the configured hostname where the RCIM master is located (see "Configuration" in Chapter 2).

-s prints the RCIM connection state

devname The device name of the desired RCIM board. Defaults to `/dev/rcim:0/rcim` which is the first RCIM board that was found on boot. `/dev/rcim:1/rcim`, `/dev/rcim:2/rcim`, etc. should be used to point this tool to the second, third, etc. RCIM board found on boot.

When interactive mode is invoked, a display similar to the following example provides configuration and status information updated every two seconds, as well as command usage. These items are explained below.

```
RCIM is isolated            RCIM version: 3
Configured RCIM master hostname is Not_Configured

Clock status and values ...
cable signal : ENABLED
tick timer   : CABLE_ENABLE LOCAL_ENABLE
posix clock  : CABLE_ENABLE LOCAL_ENABLE
tick timer   :          10.3361 seconds (        25840213 ticks)
posix clock  :       18665.4696 seconds

operations are:
s      - synchronize clocks
0[tp]  - stop clock ([t]ick/[p]osix)
1[tp]  - start clock ([t]ick/[p]osix)
w[tp]  - update clock value ([t]ick/[p]osix)
i[tp]  - isolate clock ([t]ick/[p]osix)
c[tp]  - connect clock ([t]ick/[p]osix)
d      - disable cable clock signal
e      - enable cable clock signal
q      - quit

enter operation:
```

`RCIM is` indicates the RCIM mode for this system: `master`, `pass-through slave`, `final slave` or `isolated`

`RCIM version` is the RCIM version number

`Configured RCIM master hostname is`

indicates the hostname of the RCIM master. This must be configured using the `host` configuration option (see "Configuration" in Chapter 2)

`cable signal` is one of the following:

`ENABLED/DISABLED` – For the RCIM master, indicates if the cable clock signal is being propagated to slaves. For RCIM slaves, indicates if clocks are being driven by the RCIM master or if ticking locally (without synchronization).

`CLOCK_MISSING` – Error condition indicating cable clock signal is not being properly propagated to slaves.

`CLOCK_STOPPED` – Error condition indicating cable clock signal has been stopped.

`Status:`
`tick timer`
`posix clock`
	is one of the following:
	`CABLE_SYNC` – indicates that the RCIM slave clock is being driven by RCIM master cable clock signal, if available
	`CABLE_ENABLE` – indicates that the RCIM clock resets when RCIM master does a synchronization
	`LOCAL_ENABLE` – indicates that the clock is enabled
	`NO_RESET_WHEN_DISABLED` – indicates that the clock is not reset when disabled

`Values:`
`tick timer`
`posix clock`
	current clock values for each clock

`operations are:`	This section is usage information for the interactive mode. At the `enter operation:` prompt, supply one of the operation codes to accomplish the task described for that operation. As noted, some operations require a designation for the clock to be operated on: `t` for the tick clock; `p` for the POSIX clock.

Synchronizing the Tick Clock

When the RCIM master system boots, **`rcim_clocksync`** is executed, which resets all tick clocks in the RCIM chain to zero.

When an RCIM slave system boots after the master has booted, the tick clocks on the systems in the RCIM chain will be out of sync, unless automatic clock synchronization is configured (see the section "Automatic Synchronization" below). Invoking **`rcim_clocksync`** without options on the master RCIM system synchronizes all tick clocks in the RCIM chain. See the section "The rcim_clocksync Utility" and the **`rcim_clocksync(1)`** man page for more information about this utility.

RCIM Masterclock Considerations

The RCIM POSIX clock is the system masterclock. This means that system time continually looks at the RCIM's POSIX clock and adjusts itself to match.

In previous releases of RedHawk, a broken or stopped RCIM would confuse and often lock up the system, however this is no longer the case. The RedHawk masterclock code now continually tests the RCIM POSIX clock for validity and pauses clock synchronization if problems are detected. This means that the two clocks, the RCIM POSIX clock and the system clock, become free-running. Once the RCIM POSIX clock again becomes valid clock synchronization is resumed.

In order for synchronization to resume, RedHawk must detect that both the system clock and the RCIM POSIX clock are incrementing, that each clock is incrementing at the correct rate, and that each clock has a value within approximately two seconds of the other. This latter condition is most easily achieved by executing a **`clock_settime`**(3), but other methods, as documented in **`masterclock`**(5), are available that may be more suitable for specific situations.

NOTE

See "Disabling Masterclock" on page 3-22 for information about disabling the RCIM's masterclock behavior when it is not desired.

Synchronizing the POSIX Clock

The POSIX clocks tick synchronously but generally do not have the same clock values and should not be used as a common time stamp. They can be synchronized with the other POSIX clocks in the RCIM chain so that they are consistent, if desired, by running **rcimdate** on each slave; on return the slave's RCIM POSIX clocks will be synchronized with the master.

Note that this operation also synchronizes the RCIM tick clocks. Be aware that while performing this procedure, the system time stops and time-base functions will be affected.

1. Ensure that all POSIX clocks on all RCIM slave systems are connected using the **cp** command.
2. Disable the cable clock signal on the RCIM master system (using the **d** command). The POSIX clocks on all systems should stop ticking.
3. Update the time values of the POSIX clocks on each system to the same value (using the **wp** command).
4. Re-enable the cable clock signal on the RCIM master system (using the **e** command). All clocks should begin ticking.

On RCIM systems equipped with the optional GPS module, the POSIX clock can be synchronized to standard GPS time. Refer to the section "Using GPS for System Timekeeping" below for information.

Automatic Synchronization

Each RCIM slave system can be configured so that its POSIX clock will be automatically synchronized with that of the master on boot. To do this, uncomment either the `RCIMDATE=continuous` or `RCIMDATE=oneshot` line in **/etc/sysconfig/rcim** on the slave. Subsequently, each time the slave is booted it will run **rcimdate** either once or continuously. If it is run once then the slave's POSIX clock will match that of the master until such time as some application on the master executes a **clock_settime**(3) (this is usually done by **ntp** or **ptpd**). If **rcimdate** is run in continuous mode, then the slave will execute a **clock_settime**(3) within a few seconds of each **clock_settime**(3) occurring on the master.

The system can also be configured to automatically synchronize the tick clocks when an RCIM slave system is booted. This feature is disabled by default and should be used with caution. It causes the tick clock on all systems to be reset to zero when any system in the RCIM chain is booted, which may have an undesirable impact on processes using the tick clock during synchronization.

Using GPS for System Timekeeping

On RCIM systems equipped with the optional GPS module, the POSIX clock can be used for system time synchronized to standard GPS time. A system timekeeping daemon is utilized (e.g. **ntpd** alone or **gpsd** with **chronyd**) and treats the GPS receiver as a timeserver. For information about how to configure system timekeeping daemons for this support, see the section "GPS Clock Synchronization" in Chapter 4.

NOTE

An RCIM that is configured as an RCIM slave will automatically synchronize system time to the RCIM master via the optical cable. Because of this automatic synchronization, an RCIM slave cannot also be configured to synchronize with a GPS timing source and must never enable Chronyd or NTPD.

RedHawk Linux includes the RFC-2783 pulse per second (PPS) interface that synchronizes the system time to the GPS PPS interface. The POSIX clock register is captured on the on-time edge of the GPS PPS signal. This avoids the jitter which would be introduced if an interrupt was used to capture the error between the system time and the GPS time.

A dedicated serial interface, **/dev/rcim_uart**, is used by a software daemon (e.g. **ntp** alone or **gpsd** with **chronyd**) to communicate with the GPS receiver. This symbolic link is pointed to the last RCIM found with a GPS. If another RCIM is to be used, the administrator must point this special device to the uart special device file of the desired RCIM to, for example, **/dev/rcim:3/uart**.

Likewise, the device which will receive the GPS pulse per second must be pointed to by **/dev/refclock-0**. This symbolic link also points to the last RCIM found having a GPS. If that is not the GPS that is wanted, the administrator must point this symbolic link to, for example, **/dev/rcim:3/gps**.

Tools such as **ntpq(1)**, **ntpdc(1)**, and **chronyc(1)** can be used for monitoring timekeeping activity. Other aids include various log files that can be added to **/etc/ntp.conf** or **/etc/chrony.conf** if desired. Refer to the **ntpd(1)** and **chronyd(1)** man pages for more information.

Using IRIG for System Timekeeping

On RCIM systems equipped with the optional IRIG module, the POSIX clock can be used for system time synchronized to IRIG time. The optional IRIG module allows an RCIM IV to function as an IRIG master or slave that is compliant with the IRIG B Timecode specification. For information about how to configure IRIG support, see the section “IRIG Timecode Synchronization” in Chapter 5.

NOTE

An RCIM that is configured as an RCIM slave will automatically synchronize system time to the RCIM master via the optical cable. Because of this automatic synchronization, an RCIM slave cannot also be configured to synchronize with an IRIG timing source and must never enable Chronyd or NTPD.

Interrupt Processing

One or more of the following modules is used for interrupt processing on the RCIM:

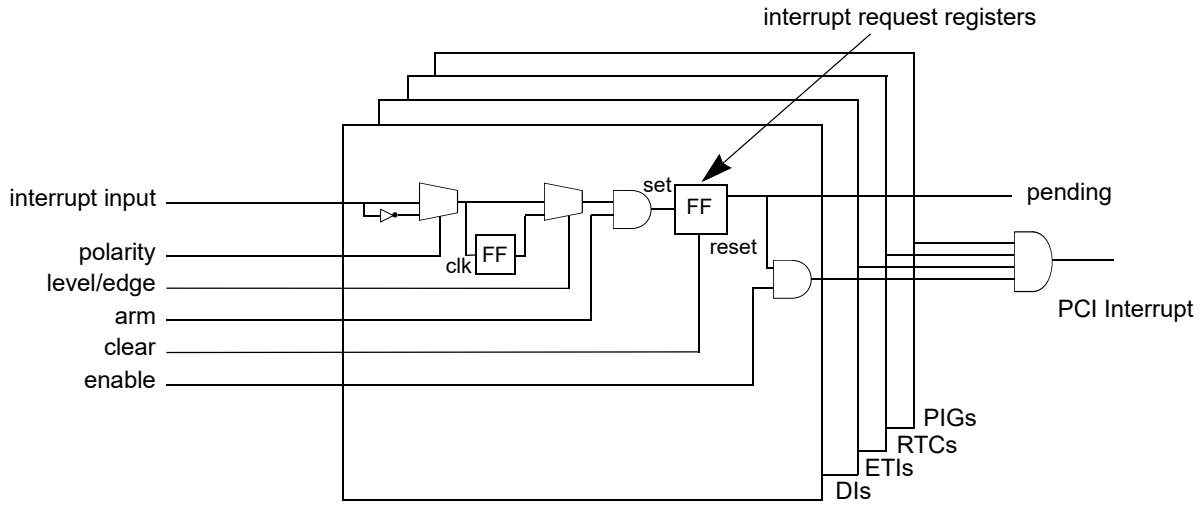
- **Edge-Triggered Interrupts (ETIs)** – An ETI allows you to use an external event to trigger an interrupt. Documentation for ETIs begins on page 3-12.
- **Real-Time Clocks (RTCs)** – An RTC allows you to set up a counter to trigger an interrupt. Documentation for RTCs begins on page 3-14.
- **External Output Interrupts** – An external output signal allows you to use any of the other signal processing modules as the source for an interrupt on an external device. Documentation for external output interrupts begins on page 3-15.
- **Programmable Interrupt Generators (PIGs)** – A PIG allows you to programmatically generate a signal that can be used to trigger interrupts. Documentation for PIGs begins on page 3-19.
- **Distributed Interrupts (DIs)** – Distributed interrupts allow you to distribute signals or interrupts across all systems connected using an RCIM chain. Documentation for DIs begins on page 3-20.

The sections that follow explain how the RCIM interrupts are processed.

Interrupt Processing Logic

Interrupt requests, whether generated by the RCIM board or by a software request, are handled by the edge-triggered interrupts (ETIs) and the distributed interrupts (DIs). Figure 3-1 illustrates how each interrupt request is processed.

Figure 3-1 Interrupt Processing Logic



DIs and ETIs must be armed and enabled for an interrupt to occur. Each interrupt can be armed/disarmed and enabled/disabled individually. After power up initialization, all interrupts are disarmed and disabled.

When the interrupt is armed, interrupt requests set a request bit. When an interrupt is disarmed, any outstanding requests are turned off and ignored.

Requests for an enabled interrupt are allowed to enter the interrupt priority resolution logic. The enable bit may be thought of as granting the RCIM board permission to deliver any interrupt requests that it accepts. When the ETI or DI is disabled, it accepts interrupt requests, but delays delivering the interrupt until it is re-enabled.

The input to the interrupt block is used to drive the DI. This output is routed to the host computer to which the RCIM board is attached, and additionally to other systems in the RCIM chain if configured to do so. The ETI or DI interrupt handler on the host clears the pending bit each time it concludes the processing of an interrupt. This clears the way for the RCIM board to output the next instance of this interrupt.

Arming and Enabling DIs and ETIs

DIs and ETIs are armed using the **`ioctl(2)`** system call with the following operations, which can be used interchangeably:

```
DISTRIB_INTR_ARM
ETI_ARM
```

DIs and ETIs are enabled using **`ioctl(2)`** with the following operations, which can be used interchangeably:

```
DISTRIB_INTR_ENABLE
ETI_ENABLE
```

Interrupt Recognition Logic

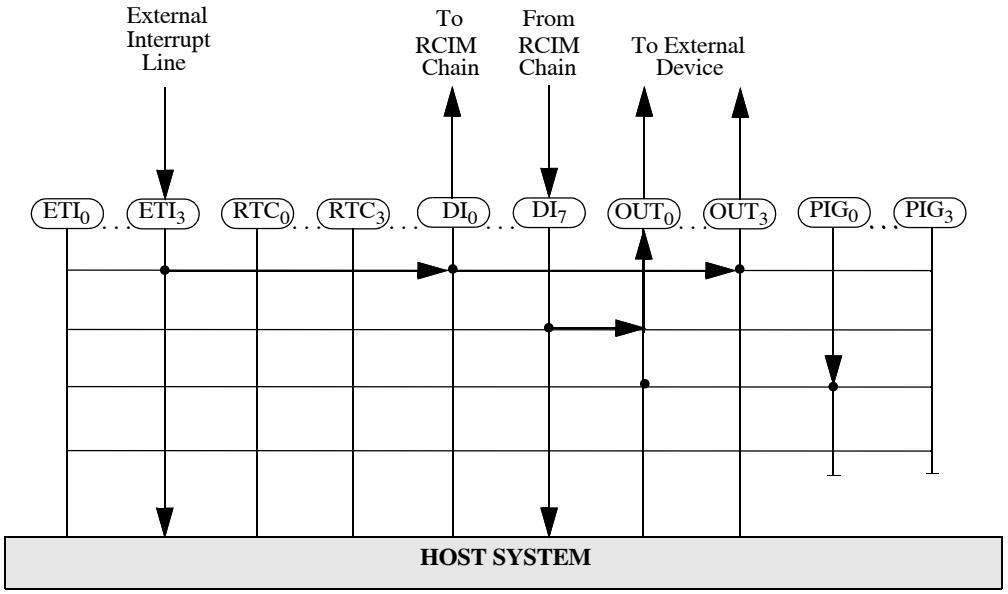
By default, the RCIM looks for the leading edge of an input signal to trigger an interrupt. Once the interrupt is recognized, it must be deasserted and reasserted to cause a new interrupt request. Optionally, an interrupt may be configured as level-sensitive. In this mode, an interrupt is triggered when the interrupt signal is high or low. Configuration options are included with the documentation for each type of interrupt later in this chapter.

For the RCIM to be able to reliably extract interrupts from an input signal, the equipment generating the signal must hold any value generated for at least 1.5 microseconds before transitioning to the reset value.

Setting up Distributed Interrupts

The RCIM provides the ability to share interrupts across interconnected systems using an RCIM chain. Although distributed interrupts are covered in detail starting on page 3-20, the figures below provide an illustration of how they operate. Guidelines for setting up distributed interrupts based on the illustration follow.

Figure 3-2 Distributed Interrupt Operation Example



In Figure 3-2, there are three scenarios:

- A signal is generated on ETI_3 that drives an interrupt on DI_0 and another on OUT_3. The interrupt is also passed to the host system.
- An interrupt is received from the RCIM chain on DI_7. This interrupt is sent to the host system and also sent to an external device via OUT_0.
- An interrupt is generated on PIG_0 that is passed to an external device via OUT_0.

Note that the local interrupt does not drive the configured DI. An ETI_REQUEST ioctl will cause a local interrupt but will not affect the DI associated with the ETI. For example, if ETI_0 is configured to drive DI_0, it will connect the external ETI input to DI_0 directly without passing through the local ETI_0 interrupt control logic. A benefit of this is that a local interrupt is not issued for every distributed PIG and ETI. A PIG should be used for a programmable software controlled interrupt and an ETI should be used for an external interrupt output.

Please note the following in this example:

- A problem can occur when more than one interrupt module tries to drive the same signal line. In the example, the ETI_3 signal drives the interrupt on

DI_0 and on OUT_3. It is possible to configure another signal processing module (say RTC_0) to drive the interrupts on DI_0 and OUT_3 at the same time. In this case, the signal that drives the line will be the one with the strongest amplifier. It is up to the administrator to avoid this condition.

- You determine which direction a distributed interrupt takes. This means that on a system where a distributed device resides, it may generate two interrupts: its local device interrupt (ETI, PIG or RTC) and the distributed interrupt. There is a separate interrupt vector for each.

  It may be desirable to receive both interrupts, but generally only one is sufficient. By disarming the distributed interrupt, one can prevent that interrupt from being generated on the local system. By default, a distributed interrupt is in a disarmed state.

Obtaining RCIM Values

There are several methods available for displaying or obtaining RCIM values:

`/proc/driver/rcim:`*N* filesystem (where *N* starts from zero):

The following files in this filesystem can be viewed (read only unless noted otherwise):
`config` – the RCIM configuration in a form suitable to cut and paste (read/write)
`interrupts` – a count of all ETI, DI and RTC interrupts per CPU and in total
`status` – miscellaneous RCIM board status and time synchronization
`rawregs` – named hex display of all readable RCIM board registers
`rtc` – status of the RTCs (run status, count values, etc.)
`eti` – status of the ETIs (armed, enabled, etc.)
`di` – status of the DI lines (armed, enabled, etc.)

`ioctl(2)` system call:

Information about a specific interrupt type can be retrieved by specifying the appropriate operation with the appropriate device file **`mmap`**'d into the address space of the program; for example, ETI_INFO with **`/dev/rcim:0/eti1`**. Refer to the **`rcim_eti(4)`**, **`rcim_rtc(4)`** and **`rcim_distrib_intr(4)`** man pages.

The RCIM_GET_INFO operation with the **`/dev/rcim:0/rcim`** device file **`mmap`**'d provides the same information as **`/proc/driver/rcim:0/config`**.

The RCIM_GET_ADDR operation with the **`/dev/rcim:0/rcim`** device file **`mmap`**'d provides the virtual and physical address of the RCIM control registers.

The header file **`/usr/include/rcim.h`** describes the layout of the information returned by RCIM_GET_INFO and RCIM_GET_ADDR. Refer to the **`rcim(4)`** man page.

`mmap(2)` system call:

`mmap` can be used to map in some or all of the device registers of the RCIM board. The register layout is in **`/usr/include/linux/rcim_ctl.h`** and in Appendix B in this guide.

Edge-Triggered Interrupts

Each RCIM board has incoming external interrupt lines, called ETIs or edge-triggered interrupts, so-named after their most common mode of operation. These lines permit users to provide their own interrupt sources. The RCIM processes and delivers these interrupts to the host system and, if they are distributed, routes and delivers them to all other RCIMs in the chain as distributed interrupts. RCIMs can support up to twelve ETIs (0-11).

Each ETI is independently configurable. An ETI may treat the incoming signal as an edge or level sensitive interrupt. If edge sensitive, it may raise an interrupt on either the rising or the falling edge. If level sensitive, it may raise interrupts for either the high or the low signal value. To specify how the incoming signal pattern is converted to interrupt requests, use one of the ETI configuration options described under "Input Configuration".

Applications in turn arm or disarm, enable or disable each ETI on each system on the fly as appropriate to the needs of those applications.

One requirement the RCIM imposes on external signal generating equipment is that the signals they output must hold any low or high signal value for at least 1 microsecond before changing to the next state. Pulses of shorter duration may not be recognized by the interrupt controller. As long as the pulse is longer than 1 microsecond, the duration of the pulse is not important.

The **`rcim_eti(4)`** man page provides complete information about ETIs.

Input Configuration

Each input (ETI) can be configured to trigger on the rising or falling edge of a signal, or on a high or low signal value using the **`input`** configuration option. This option has the following syntax:

`input`*N* **`/`**`[`**`rising`**`|`**`falling`**`|`**`high`**`|`**`low`**`]`

The default setting for an input is "falling."

The flag words (`rising`, `falling`, `high`, `low`) can be specified using the first character of the word. These words are not case sensitive.

For backwards compatibility, the original name **`eti`** is still accepted.

Examples include:

`input0/falling`	sets input 0 to trigger on the falling edge of its input signal
`input1/r`	sets input 1 to trigger on the rising edge of its input signal
`input2/h`	sets input 2 to trigger on a high signal value

See the "Configuration" section in Chapter 2 or the **`rcim(4)`** man page for the various methods available for specifying configuration options.

By default, RCIM-IV external input signals are **high true**, but RCIM-III external input signals are **low true**. Control registers exist on both to reverse the polarity if desired. External interrupt input signals have polarity control for level interrupts and falling/rising edge control for edge interrupts.

ETI Device Files

Each ETI is accessed through its own special device file:

/dev/rcim:*N*/**eti***M*

where *N* is the RCIM card number (starting from zero) and *M* is the ID of the ETI.

These files are created automatically on system boot by the **/etc/init.d/rcim** initialization script.

User Interface to ETIs

An ETI is controlled by **open(2)**, **close(2)**, and **ioctl(2)** system calls. Note that this device does not support the **read(2)**, **write(2)** or **mmap(2)** system calls.

The **open** call assigns a file descriptor to one edge-triggered interrupt and verifies that the interrupt is not currently being used by another device driver. One device file exists for each edge-triggered interrupt. A **close** call frees the file descriptor and removes any attached signals. Refer to the man pages for more information.

The following commands to **ioctl** are used to manipulate the ETIs. These commands can also be applied to DIs. All **ioctl** calls use the constants defined in **/usr/include/rcim.h**. Refer to the **rcim_eti(4)** man page for more information.

ETI_ARM	arms the ETI
ETI_DISARM	disarms the ETI
ETI_ENABLE	enables the ETI
ETI_DISABLE	disables the ETI
ETI_REQUEST	generates a software requested interrupt
ETI_INFO	gets information about the ETI
ETI_WAIT	causes the process to sleep
ETI_WAKEUP	wakes all sleeping processes
ETI_GETICNT	returns the number of times this ETI has fired
ETI_KEEPALIVE	sets or clears the keepalive state
ETI_VECTOR	gets the interrupt vector associated with ETI
IOCTLGETICNT	returns the number of times this ETI has fired (generic)
IOCTLKEEPALIVE	sets or clears the keepalive state (generic)
IOCTLVECNUM	sets the interrupt vector associated with ETI (generic)
IOCTLSIGATTACH	requests a signal when an RCIM device generates an interrupt

Note that like DIs, an ETI must be armed and enabled before an interrupt can be received.

Distributed ETIs

Any or all of the ETIs on an RCIM can be distributed to all systems connected by an RCIM chain. The source of a distributed ETI can be located on any of the RCIMs in the chain.

To determine if a specified ETI has its interrupts sent to all connected systems, use one of the methods described in the section entitled "Obtaining RCIM Values" on page 3-11. See the "Distributed Interrupts" section on page 3-20 for information about setting up distributed interrupts.

Real-Time Clocks (RTCs)

The RCIM provides real-time clock timers. Each of these counters is accessible using a special file and each can be used for almost any timing or frequency control function. Support for eight 32-bit RTCs (0-7) is provided.

The real-time clock timers are programmable to several different resolutions which, when combined with a clock count value, provide a variety of timing intervals. This makes them ideal for running processes at a given frequency (e.g., 600Hz) or for timing code segments. The timers may be one-shot or periodic; if periodic, the original load value is automatically reloaded into the counter each time zero is reached.

In addition to being able to generate an interrupt on the host system, the output of an RCIM real-time counter can be distributed to other RCIM boards for delivery to their corresponding host systems, or delivered to external equipment attached to one of the RCIM's external output interrupt lines.

The **`rcim_rtc(4)`** man page provides complete information about RTCs.

RTC Device Files

Each RTC is accessed through its own special device file:

`/dev/rcim:`*N*`/rtc`*M*

where *N* is the RCIM card number (starting from zero) and *M* is the ID of the RTC.

These files are created automatically on system boot by the **`/etc/init.d/rcim`** initialization script.

Distributed RTCs

Any or all of the RTCs on an RCIM can be distributed to all systems connected by an RCIM chain. The source of a distributed RTC may be located on any of the RCIMs in the chain.

To determine if a specified RTC has its interrupts sent to all connected systems, use one of the methods described in the section entitled "Obtaining RCIM Values" earlier in this chapter. See the "Distributed Interrupts" section on page 3-20 for information about setting up distributed interrupts.

User Interface to RTCs

A real-time clock timer is controlled by **`open(2)`**, **`close(2)`**, and **`ioctl(2)`** system calls. The **`close`** system call, if it closes the last open to the device, stops the RTC and clears its settings unless the IOCTLKEEPALIVE **`ioctl`** command was issued to the RTC before the **`close`**.

The parameters passed through **`ioctl`** control the modes of the real-time clock timer, the clock count value, and counting itself, as well as getting the current settings of the RTC. This device does not support the **`read(2)`** and **`write(1)`** system calls.

The following commands to **`ioctl`** are used to manipulate the RTCs. All **`ioctl`** calls use the constants defined in **`/usr/include/rcim.h`**. Refer to the **`rcim_rtc(4)`** man page for further information.

RTCIOCSETL	initializes RTC values (32-bit interface)
RTCIOCGETL	retrieves RTC values (32-bit interface)
RTCIOCSET	initializes RTC values (16-bit interface)
RTCIOCGET	retrieves RTC values (16-bit interface)
RTCIOCSETCNT	sets RTC clock count
RTCIOCMODCNT	modifies RTC clock count
RTCIOCGETCNT	gets RTC clock count
RTCIOCRES	gets RTC clock resolution
RTCIOCSTART	starts RTC counting
RTCIOCSTOP	stops RTC counting
RTCIOCWAIT	blocks until RTC clock count reaches zero
RTCIOCWAKEUP	wakes all sleeping processes
RTCIOCINFO	gets information about RTC
IOCTLGETICNT	returns the number of times this clock has fired
IOCTLVECNUM	sets interrupt vector for RTC
IOCTLKEEPALIVE	does not destroy the timer on final close
IOCTLSIGATTACH	requests a signal when this RTC generates an interrupt

External Output Interrupts

Each RCIM provides external output signals. These signals can be used as interrupt sources for other machines or used as signals to control external devices. RCIMs can support up to twelve external output interrupts (0-11).

The external output interrupts can be driven from one of several sources internal to the RCIM. The most common source is from the programmable interrupt generators (PIGs). PIGs provide full software control for generation of the output signals.

The pin-outs for the external interrupt connectors are described in Chapter 2.

Output Source Configuration

Each external output line can be configured to be driven by a specified source using the following configuration option:

<source> **| out***N*

The value specified for the source can be one of the following:

rtc*N*	drive the output with real-time clock timers
pig*N*	drive the output with programmable interrupt generators
input*N*	drive the output with edge triggered interrupts
eti*N*	alias for input (ETIs)
di*N*	distributed interrupts
gps	drive the output with the GPS PPS signal
irig	drive the output with the IRIG PPS signal
dcls_out	drive the output with the IRIG signal
10mhz	drive the output with the 10MHz clock
mclock	drive the output with the master time signal
none	let the interrupt output line float

Examples include:

`rtc3|out0`	sets output line 0 to be driven by real-time clock 3
`di5|out2`	sets output line 2 to be driven by distributed interrupt 5

Defaults for this configuration option are a PIG as source for the corresponding output line; i.e.:

```
pig0|out0
pig1|out1
etc.
```

NOTE

On the RCIM IV you cannot specify a pin as an input source to drive the same pin's output. For example, trying to set "**`input0|out0`**" is invalid and will result in an "Invalid Argument" error.

See the "Configuration" section in Chapter 2 or the **`rcim(4)`** man page for the various methods available for specifying configuration options.

Note that on the RCIM IV each EXT_INOUT pin needs to first be configured as an output line before it can be driven with a source. See "RCIM IV Pin Configuration" on page 3-23 for more information.

By default, RCIM-IV external output signals are **high true**, but RCIM-III external output signals are **low true**. Control registers exist on both to reverse the polarity if desired.

MSI-X Interrupt Configuration

The RCIM IV fully supports the MSI-X standard and allows attaching sources to up to 16 different interrupts on compatible hardware. Each MSI-X interrupt can be given its own unique core affinity, eliminating interference from other RCIM sources and delivering the lowest interrupt latencies possible on the current hardware.

NOTE

For backward compatibility with older versions of RedHawk, by default only one MSI-X interrupt vector is enabled (`irq0`) which is used for all MSI-X interrupts.

To enable additional MSI-X interrupt vectors, boot the RedHawk kernel with the `rcim.max_msix_irqs=`*N* boot option, where *N* can be set to any integer between 1 and 16.

To fully disable all MSI-X support, boot the RedHawk kernel with the `rcim.nomsix` boot option.

MSI-X Source Configuration

Each MSI-X interrupt can be configured to be associated with a specified source using the following configuration option:

<source> **| irq***N*

The value specified for the source can be one of the following:

rtc*N*	associate the real-time clock with the specified IRQ
input*N*	associate the input with the specified IRQ
eti*N*	alias for input (ETIs)
di*N*	associate the distributed interrupt with the specified IRQ
gps	associate the GPS PPS signal with the specified IRQ
irig	associate the IRIG PPS signal with the specified IRQ
none	ensure no source is associated with the specified IRQ

Examples include:

`rtc3|irq3`	associates real-time clock 3 interrupts with IRQ 3
`input8|irq4`	associates input 8 interrupts with IRQ 4

Defaults for MSI-X IRQ source associations are configured as follows:

```
rtc0|irq1
rtc1|irq2
rtc2|irq3
rtc3|irq4
rtc4|irq5
rtc5|irq6
rtc6|irq7
```

```
rtc7|irq8
input6|irq9
input7|irq10
input8|irq11
input9|irq12
input10|irq13
input11|irq14
di0|irq15
```

Note that `irq0` is not mentioned above because it is a general purpose IRQ that services all source interrupts that have not been explicitly associated with another IRQ.

A source can be explicitly associated with `irq0` to remove any previous association it had with a different IRQ. In addition, sources will be automatically associated with `irq0` whenever a configuration change results in the source no longer being explicitly associated with a non-zero IRQ.

MSI-X Compatibility

Existing shell scripts or programs that process the contents of the **`/proc/interrupts`** file to discover information about RCIM interrupts may need to be modified to handle the additional RCIM interrupt lines that will be present when more than one MSI-X interrupt vector is allocated.

By default, with only one interrupt vector allocated for all MSI-X interrupts, the single RCIM line in the **`/proc/interrupts`** file will be unchanged from previous releases. For example, the RCIM line should look similar to the following line on a quad-core system:

```
242:   0   0   0   0  IR-PCI-MSI 49283072-edge     rcim
```

However, if MSI-X is configured with more than one interrupt vector, (e.g., when the kernel is booted with `rcim.max_msix_irqs=16`), then the contents of the **`/proc/interrupts`** file will be significantly different and there will be one line for each interrupt vector. For example, the RCIM lines should look similar to the following lines on a quad-core system:

```
242:   0   0   0   0  IR-PCI-MSI 49283072-edge     rcim-irq0
249:   0   0   0   0  IR-PCI-MSI 49283073-edge     rcim-irq1
250:   0   0   0   0  IR-PCI-MSI 49283074-edge     rcim-irq2
251:   0   0   0   0  IR-PCI-MSI 49283075-edge     rcim-irq3
252:   0   0   0   0  IR-PCI-MSI 49283076-edge     rcim-irq4
253:   0   0   0   0  IR-PCI-MSI 49283077-edge     rcim-irq5
254:   0   0   0   0  IR-PCI-MSI 49283078-edge     rcim-irq6
255:   0   0   0   0  IR-PCI-MSI 49283079-edge     rcim-irq7
256:   0   0   0   0  IR-PCI-MSI 49283080-edge     rcim-irq8
257:   0   0   0   0  IR-PCI-MSI 49283081-edge     rcim-irq9
258:   0   0   0   0  IR-PCI-MSI 49283082-edge     rcim-irq10
259:   0   0   0   0  IR-PCI-MSI 49283083-edge     rcim-irq11
260:   0   0   0   0  IR-PCI-MSI 49283084-edge     rcim-irq12
261:   0   0   0   0  IR-PCI-MSI 49283085-edge     rcim-irq13
262:   0   0   0   0  IR-PCI-MSI 49283086-edge     rcim-irq14
263:   0   0   0   0  IR-PCI-MSI 49283087-edge     rcim-irq15
```

NOTE

The examples above are representative of systems that have only one RCIM card installed. If more than one RCIM card is present, the interrupt names will appear as `RCIM:`*`R`*`-irq`*`N`* where *R* is the consecutive number of the RCIM card (starting from 0) and *N* is the associated MSI-X IRQ vector number.

Programmable Interrupt Generators (PIGs)

Each RCIM provides programmable interrupt generators (PIGs). PIGs are generally used to provide software control for the output of an external output signal. Additionally, the PIGs can be used to drive a signal onto a distributed interrupt line, permitting user software to generate distributed interrupts that are simultaneously delivered to all RCIMs in an RCIM chain. RCIM III and RCIM II support twelve PIGs (0-11); RCIM I supports four PIGs (0-3).

The **`rcim_pig(4)`** man page provides complete information about PIGs.

By default, RCIM-IV external output signals are **high true**, but RCIM-III external output signals are **low true**. Control registers exist on both to reverse the polarity if desired. Programmable interrupt generator (PIG) outputs can be set or reset to any desired state.

PIG Device File

The device file **`/dev/rcim:`***`N`***`/pig`** is used to access the PIG register on the local system. This file must be mapped into the address space of a program using **`mmap(2)`**. By default, only users with root privileges have access to do this.

On RCIM III and RCIM IV, the PIG register is 12 bits wide, one bit for each PIG. Two additional registers allow the PIG bits to be set and cleared in a multiprocessor safe manner. In the **`mmap`**'ed PIG register page, the set register (PIGS) is at offset 0x10 and the clear register (PIGC) is at offset 0x20. Setting a PIG generates a distributed interrupt or external output, depending on how the PIGs are connected. The required length of the signal depends upon the requirements of the attached device.

If the signal is being fed into an RCIM IV, it must hold any low or high value for at least 1 microsecond before changing to the next state. For an RCIM III the duration held must be 1.5 microseconds.

Distributed PIGs

Any or all of the PIGs on an RCIM can be distributed to all systems connected by an RCIM chain. The source of a distributed PIG may be located on any of the RCIMs in the chain.

To determine if a specified PIG has its interrupts sent to all connected systems, use one of the methods described in the section entitled "Obtaining RCIM Values" earlier in this chapter. See the "Distributed Interrupts" section below for information about setting up distributed interrupts.

Distributed Interrupts

The real heart and power of the RCIM lies in its distributed interrupt system. Each RCIM can distribute interrupts simultaneously to all systems connected via an RCIM chain. RCIMs can support up to twelve distributed interrupts (0-11). A diagram of this functionality and guidelines for setting up distributed interrupts can be found in the section "Setting up Distributed Interrupts" earlier in this chapter.

Any of the edge-triggered interrupts, real-time clock timers, GPS pulse per second or programmable interrupt generators on any of the RCIM boards in the chain can be configured to be distributed. A distributed device file is associated with each of the distributed interrupts.

RCIM distributed interrupts must be configured on each system attached to the RCIM that is either broadcasting or intending to receive a distributed interrupt. Distributed interrupts can also be configured and used locally on an isolated system. Configuration details are given in the "DI Configuration" section below. See the section "Obtaining RCIM Values" on page 3-11 for the methods available for obtaining configuration information.

The **`rcim_distrib_intr(4)`** man page provides complete information about DIs.

DI Configuration

It is important that all RCIM-connected systems have a compatible configuration for the distributed interrupt lines of the RCIM.

By default, no distributed interrupts are configured.

Distributed interrupts must first have a source, and then can be configured to trigger on the rising or falling edge of a signal, or on a high or low signal value.

To define the source for a distributed interrupt, use the following configuration option:

<source> **`|`** **`di`***N*

The value specified for the source can be one of the following:

rtc*N*	real-time clock timers
pig*N*	programmable interrupt generators
input*N*	edge-triggered interrupts
eti*N*	alias for edge-triggered interrupts
gps	GPS signal
irig	IRIG signal
none	the RCIM is not to drive this distributed interrupt

Examples include:

`rtc3|di6`	sets distributed interrupt 6 to be driven by real-time clock 3
`pig1|di3`	sets distributed interrupt 3 to be driven by programmable interrupt generator 1
`none|di0`	the RCIM is not to drive distributed interrupt 0

Each distributed interrupt can be configured to trigger on the rising or falling edge of a signal, or on a high or low signal value using the **di** configuration option. This option has the following syntax:

di*N***/**[**rising**|**falling**|**high**|**low**]

The flag words (`rising`, `falling`, `high`, `low`) can be specified using the first character of the word. These words are not case sensitive.

Examples include:

`di0/falling`	sets distributed interrupt 0 to trigger on the falling edge of its input signal
`di1/r`	sets distributed interrupt 1 to trigger on the rising edge of its input signal

See the "Configuration" section in Chapter 2 or the **rcim(4)** man page for the various methods available for specifying configuration options.

DI Device Files

Each distributed interrupt is accessed through its own special device file:

/dev/rcim:*N***/di***M*

where *N* is the RCIM card number (starting from zero) and *M* is the ID of the distributed interrupt.

These files are created automatically on system boot by the **/etc/init.d/rcim** initialization script.

User Interface to DIs

A distributed interrupt is controlled by **open(2)**, **close(2)**, and **ioctl(2)** system calls. Note that this device does not support the **read(2)**, **write(2)** and **mmap(2)** system calls.

The **open** call assigns a file descriptor to one distributed interrupt. A **close** call frees the file descriptor and, if it is the last close, disarms the interrupt if the IOCTLKEEPALIVE state is clear. Refer to the man pages for more information.

The following commands to **ioctl** are used to manipulate distributed interrupts. These commands can also be applied to ETIs. All **ioctl** calls use the constants defined in **/usr/include/rcim.h**. Refer to the **rcim_distrib_intr(4)** man page for more information.

DISTRIB_INTR_ARM	arms the DI
DISTRIB_INTR_DISARM	disarms the DI
DISTRIB_INTR_ENABLE	enables the DI
DISTRIB_INTR_DISABLE	disables the DI
DISTRIB_INTR_REQUEST	generates a software requested interrupt
DISTRIB_INTR_INFO	gets information about the DI

DISTRIB_INTR_WAIT	sleeps until the next DI
DISTRIB_INTR_WAKEUP	wakes up all sleeping processes
DISTRIB_INTR_KEEPALIVE	sets or clears keepalive state
DISTRIB_INTR_GETICNT	returns the number of times this DI has fired
DISTRIB_INTR_VECTOR	gets interrupt vector for the DI
IOCTLKEEPALIVE	sets or clears keepalive state (generic)
IOCTLGETICNT	returns the number of times this DI has fired (generic)
IOCTLVECNUM	sets interrupt vector for the DI (generic)
IOCTLSIGATTACH	requests a signal when an RCIM device generates an interrupt

Note that like ETIs, a distributed interrupt must be armed and enabled before an interrupt can be received.

Disabling Masterclock

By default, every RCIM card is automatically registered as a masterclock at the time it is discovered and initialized by the RCIM driver, which means that the RCIM's time will be used as the basis for system time. The masterclock behavior can be enabled and disabled with the following configuration syntax:

[**clock** | **noclock**]

Disabling the RCIM's default masterclock behavior may be necessary under certain circumstances, for example, when it is desired that a 3rd party GPS card be used as the basis for system time.

For more information, see "RCIM Masterclock Considerations" on page 3-5 and the **masterclock**(5) man page.

External Clock Input

RCIMs can optionally be driven by an external 10 MHz clock signal. The external clock signal will be used as the RCIM's primary clock source for the RTCs, the tick clock, and the POSIX clock. This external clock source replaces the internal crystal of the RCIM.

For the RCIM III, the external clock input signal must be connected to the dedicated EXT_CLKIN input pin (see "External Interrupt I/O Connector" on page 2-11).

For the RCIM IV, the external clock input signal can be connected to any EXT_INOUT pin (see "External Interrupt I/O Connector" on page 2-5) that has been configured to receive the external clock input signal using the following configuration syntax:

pin*N* **| ext_clock_in**

For backward compatibility, `mclock_in` is accepted as an alias for `ext_clock_in`.

Note that the specific pin used to receive the external clock input signal must also be configured as an input, as described in the following section.

RCIM IV Pin Configuration

Unique to the RCIM IV, each of the EXT_INOUT pins can be configured to be either input or output, as well as terminated or non-terminated.

pin*N***/**[**out**|**in**]**/**[**non-terminated**|**terminated**]

The flag words (`out`, `in`, `non-terminated`, `terminated`) can be specified using the first character of the word. These words are not case sensitive.

Examples include:

`pin0/out/n`	configure EXT_INOUT0 to be output with no termination
`pin1/in/t`	configure EXT_INOUT1 to be input with termination
`pin2/i/terminated`	configure EXT_INOUT2 to be input with termination
`pin3/o/non-terminated`	configure EXT_INOUT3 to be output with no termination

Default: EXT_INOUT pins 0-5 are outputs, EXT_INOUT pins 6-11 are inputs, and all pins are non-terminated.

Pins must be configured as inputs before they can be used as sources, and pins must be configured as outputs before they can be driven by sources.

NOTE

On the RCIM IV you cannot specify a pin as an input source to drive the same pin's output. For example, trying to set "**input0|out0**" is invalid and will result in an “Invalid Argument” error.

See the “Configuration” section in Chapter 2 or the **rcim(4)** man page for the various methods available for specifying configuration options.

4
GPS Clock Synchronization

This chapter describes how to configure GPS clock synchronization with RCIMs that include the optional GPS module.

Overview

The optional GPS module can be utilized by system timekeeping daemons to synchronize the RCIM's POSIX clock with GPS time. This feature allows all RCIMs in a chain to be synchronized with GPS time from the RCIM master.

If your RCIM contains the optional GPS module, system timekeeping daemons must be installed and configured to use the GPS receiver to synchronize the RCIM's POSIX clock to GPS time. RedHawk supports two different synchronization methods:

1. Using the GPSD and Chronyd daemons.
2. Using the NTPD daemon.

Both methods will be described in the following main sections.

NOTE

The RCIM-IV GPS module operates in overdetermined clock mode from a fixed antenna location. If mobile use is required, contact Concurrent Support at (800) 245-6453 or via this URL: **https://www.concurrent-rt.com/support/**

GPSD and Chronyd

RedHawk versions 7.x and 8.x support GPS clock synchronization utilizing the **`chronyd`** daemon along with the **`gpsd`** daemon. The following sections provide details for configuring, running and then verifying GPS clock synchronization using these daemons.

NOTE

An RCIM that is configured as an RCIM slave will automatically synchronize system time to the RCIM master via the optical cable. Because of this automatic synchronization, an RCIM slave cannot also be configured to synchronize with a GPS timing source and must never enable Chronyd or NTPD.

Configuring gpsd

Follow these steps to configure **gpsd** to synchronize the RCIM clock with GPS time:

1. Verify that the `gpsd` rpm is installed on the system:

   ```
   # rpm -q gpsd
   ```

   If the package is not already installed, the necessary **gpsd** packages can be found in these locations:

   - the RedHawk Linux Installation Disc in the `gpsd` directory
   - via Concurrent's online repositories utilizing the NUU update utility
   - the Extra Packages for Enterprise Linux (EPEL) online repositories

   NOTE

   Versions of **gpsd** prior to 3.23 contain a serious bugs that can affect proper timekeeping. Please make sure to use a version of **gpsd** that is at version 3.23 or newer. Contact Concurrent Support if you have trouble locating a proper version.

   NOTE

   On systems installed with Ubuntu, the **gpsd** packages are available from the standard built-in distribution repositories. Ubuntu **gpsd** setup requires the same configuration files described here.

   If you have access to the RedHawk Linux Installation Disc, simply mount the disc and run the following command as the root user from the **gpsd** directory on the disc:

   ```
   # rpm -Uvh ./*.rpm
   ```

   Once the installation completes, skip to section 2 below.

   If you do not have access to the RedHawk Linux Installation Disc, then you can still find the software in the Extra Packages for Enterprise Linux (EPEL) repository located at the following URL:

   `https://fedoraproject.org/wiki/EPEL`

   First, download and install the `epel-release` RPM for your RedHawk version so that you can easily install EPEL packages via **yum** with automatic dependencies.

   Also make sure that your base distribution repositories (e.g. BaseOS and AppStream) are enabled for dependencies. See the **yum(8)** man page for more information.

   Once the `epel-release` RPM is installed, issue the following command to install the **gpsd** daemon and clients:

   ```
   # yum install gpsd gpsd-clients
   ```

2. Edit the **/etc/sysconfig/gpsd** file on the system and ensure that the following variables are set as indicated below:

```
OPTIONS="-n"
DEVICES="/dev/rcim_uart0"
BAUDRATE="9600"
```

These variable settings are required for GPS clock synchronization on the RCIM.

NOTE

The versions of **gpsd** for RedHawk 7.x only require the `OPTIONS` variable to specify the device name, for example, `OPTIONS="-n /dev/rcim_uart"`.

Other variable settings that exist in the file can be left unchanged (e.g. `USBAUTO`).

3. Verify that the **gpsd** daemon is enabled and running.

   On newer versions of RedHawk that utilize **systemd**:

   Run the following command to check the status of **gpsd**:

   ```
   # systemctl | grep gpsd
   gpsd.service
       loaded active running    GPS (...) daemon
   gpsd.socket
       loaded active running    GPS (...) daemon socket
   ```

   If the **gpsd** service is not running, issue the following commands as the root user to enable and start the daemon:

   ```
   # systemctl enable gpsd
   # systemctl start gpsd
   ```

   On older versions of RedHawk without **systemd**:

   Run the following command to check the status of **gpsd**:

   ```
   # service gpsd status
   gpsd (pid 6031) is running...
   ```

   If the **gpsd** service is not running, issue the following commands as the root user to enable and start the daemon:

   ```
   # chkconfig gpsd on
   # service gpsd start
   Starting gpsd:          [  OK  ]
   ```

Always make sure to restart **gpsd** or reboot the system after making changes to the `/etc/sysconfig/gpsd` configuration file.

Configuration of **gpsd** is the first step, but **chronyd** must also be configured before GPS clock synchronization can be achieved.

Configuring chronyd

Follow these steps to configure **chronyd** for GPS clock synchronization:

1. Verify that the `chrony` rpm is installed on the system:

   ```
   # rpm -q chrony
   ```

   If it is not installed, install in from the base distribution Updates DVD that came with your system, or you can install it over the Internet with the following command:

   ```
   # yum install chrony
   ```

2. Add the following lines to the **`/etc/chrony.conf`** file on the system:

   ```
   refclock PPS /dev/pps0 lock NMEA refid RCIM
   refclock SHM 0 refid NMEA noselect
   ```

   These lines are required for GPS clock synchronization on the RCIM.

   If multiple GPS-capable devices are present, select the `/dev/pps#` of the device desired for GPS clock synchronization.

   NOTE

   The above two `refclock` lines must be the only `refclock` lines that exist in the **`/etc/chrony.conf`** file, ignoring any instances in commented out lines. In addition, the system must not run any other NTP or PTP daemons to synchronize the time using a network-based time source. Ensure that any installed **`ntpd`** and **`ptpd`** services are disabled and stopped.

   NOTE

   The `refclock SHM` line described above will work for most systems, but some systems may require the addition of an offset. For example, the following line specifies that a ± 400 millisecond offset is required for **`chronyd`** to syncronize correctly:

   ```
   refclock SHM 0 offset 0.4 refid NMEA noselect
   ```

   See "Calculating PPS time offsets" on page 4-5 for instructions to determine the correct offset required if **`chronyd`** fails to synchronize with the default `refclock SHM` line on your system.

3. Verify that the **`chronyd`** daemon is enabled and running.

   NOTE

   `gpsd` must be configured correctly and running before **`chronyd`** can be started successfully for GPS clock synchronization. See "Configuring gpsd" on page 4-2 for more information.

   On newer versions of RedHawk that utilize **`systemd`**:

   Run the following command to check the status of **`chronyd`**:

```
# systemctl | grep chronyd
chronyd.service
        loaded active running   NTP client/server
```

If the **chronyd** service is not running, issue the following commands as the root user to enable and start the daemon:

```
# systemctl enable chronyd
# systemctl start chronyd
```

On older versions of RedHawk without **systemd**:

Run the following command to check the status of **chronyd**:

```
# service chronyd status
chronyd (pid 12473) is running...
```

If the **chronyd** service is not running, issue the following commands as the root user to enable and start the daemon:

```
# chkconfig chronyd on
# service chronyd start
Starting chronyd:       [  OK  ]
```

Always make sure to restart **chronyd** or reboot the system after making changes to the `/etc/chrony.conf` configuration file.

At system power-on, once the RCIM's GPS receiver has locked onto GPS satellites and all data is received, accurate timekeeping will be available. The initial cold-boot GPS synchronization process can require up to 15 minutes to complete, though it normally completes within one or two minutes when using the more modern GPS modules on the RCIM IV and the latest generation RCIM III. In all cases, GPS synchronization after a warm reboot should complete within one or two minutes.

Calculating PPS time offsets

The `refclock SHM` configuration line in **/etc/chrony.conf** may require a PPS time offset correction to be manually configured before **chronyd** will synchronize. For example, the following line specifies that a -300 millisecond offset correction is required for **chronyd** to synchronize correctly on the current system:

```
refclock SHM 0 offset -0.3 refid NMEA noselect
```

Offset corrections can be required because a pulse-per-second (PPS) reference clock always needs a non-PPS source to determine which second of UTC corresponds to each pulse, and for both times to be correlated they must be within ± 200 milliseconds. Non-PPS time is supplied via the RCIM GPS UART (NMEA) and various hardware factors can cause the two time sources to be offset by more than 200 milliseconds, thus requiring an offset correction to be manually specified.

NOTE

Older RCIM III cards with older GPS modules may consistently synchronize to a time that is one second off from the reference clock; for these cards, manually adding a correction offset to /etc/chrony.conf is also required.

To determine the offset correction required, run the following command before any `offset` keyword has been placed in the `refclock SHM` configuration line:

```
# chronyc sources
```

Normally, you should see `Last sample` values similar to the following:

```
210 Number of sources = 2
MS Name/IP address   ...   Last sample
==================== ... ===============================
#* RCIM               ...     -0ns[   +0ns] +/-  101ns
#? NMEA               ...    +77ms[  +77ms] +/- 1737us
```

The `+77ms` sample in the above NMEA line indicates that the PPS and non-PPS time sources are correlated to under ± 200 milliseconds, allowing **`chronyd`** to synchronize to the RCIM GPS, as evidenced by the * status shown for the RCIM.

However, on some systems, you may see `Last sample` values that are greater than ± 200 milliseconds, similar to the following:

```
210 Number of sources = 2
MS Name/IP address   ...   Last sample
==================== ... ===============================
#? RCIM               ...     -0ns[   +0ns] +/-  101ns
#? NMEA               ...   +574ms[ +574ms] +/- 1558us
```

The `+574ms` sample in the above NMEA line indicates that the PPS and non-PPS time sources are not correlated to under ± 200 milliseconds, preventing **`chronyd`** from synchronizing and causing the `?` status to be shown for the RCIM. To fix this problem, place a -500 millisecond offset correction on the `refclock SHM` line for this system:

```
refclock SHM 0 offset -0.5 refid NMEA noselect
```

Then invoke the following commands to restart **`chronyd`** and re-examine the adjusted NMEA last sample values to ensure they are now within ± 200 milliseconds:

```
# systemctl restart chronyd
# chronyc sources
```

NOTE

For offsets of 500 milliseconds and larger, the last sample values displayed may not accurately represent the true direction of the time correlation's error. In this case, you should add a negative offset correction (`-0.5`) and then a positive correction (`0.5`) to determine which value correctly allows **`chronyd`** to synchronize.

For more information about offsets, see the Chrony FAQ at the following URL: **https://chrony.tuxfamily.org/faq.html#_using_a_pps_reference_clock**

Verifying GPS Operation with chronyc

To determine when the GPS is producing accurate system time, use the source listing feature of **`chronyc(1)`** as shown below. This example utilizes command line options, however **`chronyc`** can also be run interactively. Refer to the man page for complete information.

```
# chronyc sources
210 Number of sources = 3
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
#* RCIM                          0   4   377    14    -15ns[  -16ns] +/-  294ns
#? NMEA                          0   4   377    11    -82ms[  -82ms] +/-  100ms
^- ntp1.sample.com              1   6   377    41    +12us[  +12us] +/- 1455us
```

The output shows how the system time compares to other time sources. This includes the GPS receiver and other timeservers.

The `Name/IP address` column indicates the hostname of the timeserver. The `RCIM` timeserver represents the snapshot of the POSIX time when the GPS received the last PPS signal. The `NMEA` timeserver represents the delayed time when **`chronyd`** was able to read the PPS from the GPS via the UART. Other timeservers have been either locally defined or automatically assigned by pool.ntp.org.

The `M` column indicates the mode of the source, with '`^`' indicating an external timeserver, '`=`' indicating a peer, and '`#`' indicating a locally connected reference clock.

The `S` column indicates the state of the source, with '`*`' indicating the source to which is currently synchronized, '`+`' indicating sources which are being combined with the synchronized source, '`-`' indicating sources which are being excluded by the combining algorithm, '`?`' indicating sources that are not acceptable or accessible, and both '`x`' and '`~`' indicating a source that has been deemed untrustworthy.

The `Stratum` column indicates the stratum number. A stratum zero system should have a direct connection to an authoritative source.

The `Poll` column indicates how frequently this server is being polled, expressed as a base-2 logarithm of the interval. For example, 4 indicates polling occurs every 16 every seconds, and 6 indicates polling occurs every 64 seconds.

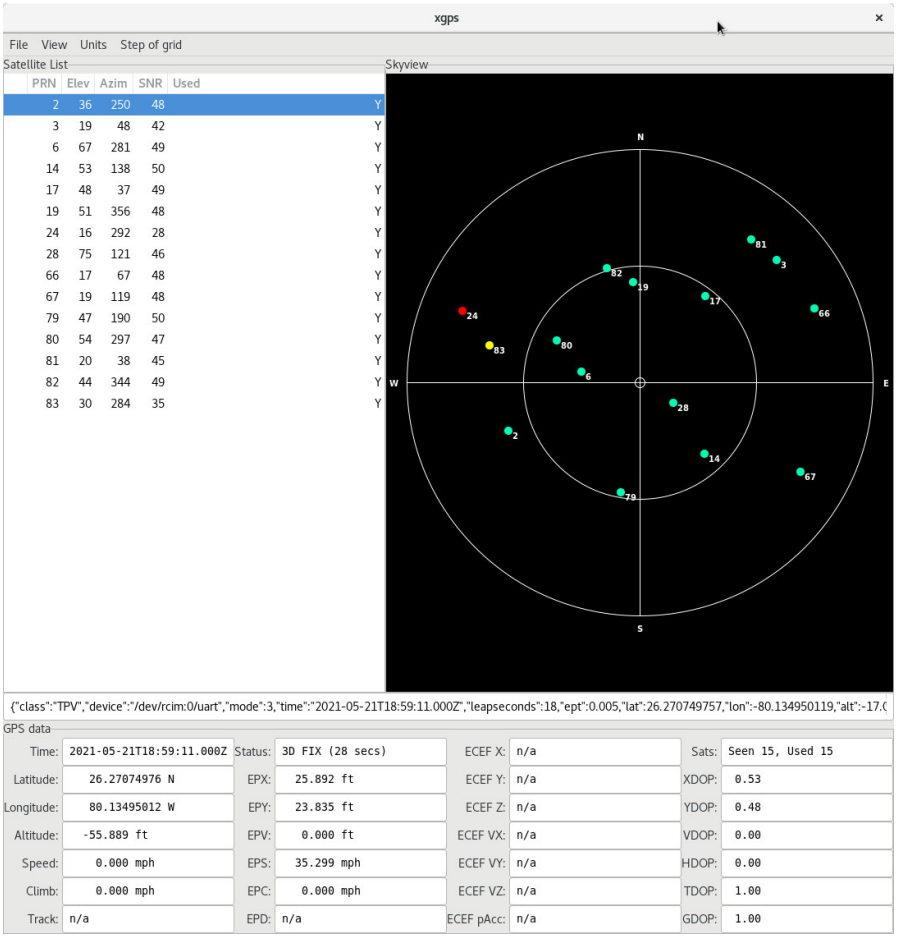
The `Reach` column indicates the source's reachability register printed as an octal number. A value of 377 indicates that valid replies were received for the last eight transmissions.

The `LastRX` column indicates how long ago the last good sample (which is shown in the next column) was received from the source.

The `Last sample` column indicates the offset between the local clock and the source at the last measurement. The number in the square brackets shows the actual measured offset. The number to the left of the square brackets shows the original measurement, adjusted to allow for any slews applied to the local clock. The number following the +/- indicator shows the margin of error in the measurement. Positive offsets indicate that the local clock is ahead of the source.

Viewing GPS Satellites with xgps

The **`xgps`** utility is provided by the `gpsd-clients` RPM. **`xgps`** displays the current GPS position and time along with the locations of accessible satellites. The following screenshot shows an example of the **`xgps`** display:



In the Skyview pane, satellites are color-coded to indicate quality of signal, with light greens indicating higher qualities and dark reds indicating lower qualities; consult the data display to the left for exact figures in dB.

Circles indicate GPS satellites. Filled circles indicate that the satellite was used during the most recent GPS calculation, and outlined circles indicate satellites that were not used.

Hovering over a symbol with the mouse pointer will temporarily display a small popup window with satellite details from the Satellite List pane.

NTPD

Older versions of RedHawk and most newer versions support GPS clock synchronization utilizing the legacy **ntpd** daemon. The following sections provide details for configuring, running and then verifying GPS clock synchronization using **ntpd**.

Configuring ntpd

Follow these steps to configure **ntpd** for GPS clock synchronization:

1. Verify that the `ccur-ntp` rpm is installed on the system:

   ```
   # rpm -q ccur-ntp
   ```

   If it is not installed, refer to the *RedHawk Linux Release Notes* for instructions to install this package from the *RedHawk NTP w/ RCIM GPS* installation media.

2. The file **/etc/ntp.conf** supplied with the `ccur-ntp` rpm contains the following lines that are required to use the GPS.

   ```
   server 127.127.8.0 mode 138 prefer  #PARSE TSIP (10)+ PPS(128)
   fudge 127.127.8.0 flag3 1           #enable PPS signal
   ```

   No changes to these lines are required for proper GPS synchronization, however more information about the specified parameters can be found at these URLs:

   ```
   http://doc.ntp.org/4.1.2/clockopt.htm
   http://doc.ntp.org/4.1.2/driver8.htm
   ```

   The following three lines define a pool of world-wide servers that are randomly selected at power-on for time synchronization. This feature acts as the default NTP configuration and serves as backup to GPS. You may wish to include your local country code before “pool” in these entries for best results; e.g., `0.us.pool.ntp.org`. See **www.pool.ntp.org** for more information.

   ```
   server 0.pool.ntp.org
   server 1.pool.ntp.org
   server 2.pool.ntp.org
   ```

   A block of commented out entries beginning with “logfile” is used to configure files used for logging statistics. Uncomment the entries if you wish to enable them.

   In addition to the log files, **ntpq(1)** and **ntpdc(1)** are used for NTP monitoring. For more information about NTP, refer to the **ntpd(1)** man page and **www.ntp.org**.

3. Verify that the **ntpd** daemon is enabled and running.

On newer versions of RedHawk that utilize **systemd**:

Run the following command to check the status of **ntpd**:

```
# systemctl | grep ntpd
ntpd.service
        loaded active running   Network Time Service
```

If the **ntpd** service is not running, issue the following commands as the root user to enable and start the daemon:

```
# systemctl enable ntpd
# systemctl start ntpd
```

On older versions of RedHawk without **systemd**:

Run the following command to check the status of **ntpd**:

```
# service ntpd status
ntpd (pid 8537) is running...
```

If the **ntpd** service is not running, issue the following commands as the root user to enable and start the daemon:

```
# chkconfig ntpd on
# service ntpd start
Starting ntpd:                    [  OK  ]
```

Always make sure to restart **ntpd** or reboot the system after making changes to the `/etc/ntp.conf` configuration file.

At system power-on, once the RCIM's GPS receiver has locked onto GPS satellites and all data is received, accurate timekeeping will be available. The initial cold-boot GPS synchronization process can require up to 15 minutes to complete, though it normally completes within one or two minutes when using the more modern GPS modules on the RCIM IV and the latest generation RCIM III. In all cases, GPS synchronization after a warm reboot should complete within one or two minutes.

NOTE

The **gpsd** and **chronyd** daemons achieve GPS synchronization consistently faster than **ntpd**. See "GPSD and Chronyd" on page 4-1

Verifying GPS Operation with ntpq

To determine when the GPS is producing accurate system time, use the peer listing feature of **ntpq(1)** as shown below. This example utilizes command line options, however **ntpq** can also be run interactively. Refer to the man page for complete information.

```
# /usr/sbin/ntpq -np
remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
xns2.medbanner.c 192.43.244.18    2 u   33   64  377   72.443   -2.897   11.235
+toshi.keneli.or .GPS.            1 u   43   64  377   27.915    0.938    2.075
```

```
-216.56.81.86       193.131.101.50   3   u   33    64    377    49.388   -0.579    2.710
+new.localdomain    .GPS.            1   u   42    64    377    0.182    0.010     0.020
*GENERIC(0)         .GPS.            0   l   46    64    377    0.000    0.000     0.001
```

The output shows how the system time compares to other time sources. This includes the GPS receiver and other timeservers.

The column labeled `remote` is the hostname of the timeserver. The system `new.localdomain` is a local network timeserver; `GENERIC(0)` is the GPS attached to the RCIM. The other lines are timeservers assigned by pool.ntp.org. The first column indicates which servers are being selected for synchronization. The '*' in front of `GENERIC(0)` indicates that the RCIM GPS receiver is being used as the system peer.

The columns `delay`, `offset`, and `jitter` are all times in milliseconds. The offset is the difference between the local system time and the time source. In this case we are synchronized to the GPS receiver to microsecond accuracy.

The `delay` field is the measured network delay to exchange the time with the remote server.

The `jitter` measures the difference between offset values from the same source.

The `refid` indicates where the remote system gets its time.

The `st` column is the stratum number. A stratum zero system should have a direct connection to an authoritative source.

The `poll` column shows how frequently this server is being polled. The `when` column is the time in seconds since the last poll.

The `reach` column is a bitmap in octal which shows if recent polls have been successful. The value 377 would indicate that the last 8 polls succeeded.

GPS Synchronization Accuracy

During synchronized GPS operation, the RCIM POSIX clock should be continuously synchronized to the GPS PPS signal. To verify accuracy, synchronization accuracy is measured once per second and the values for the last two hours of operation are temporarily recorded in the **`/proc/driver/rcim/gps-stats`** file.

NOTE

On RedHawk 7.5 and earlier only one hour of history is recorded.

The accuracy measured is the difference in nanoseconds between the RCIM POSIX clock (the basis for system time) and the GPS PPS signal.

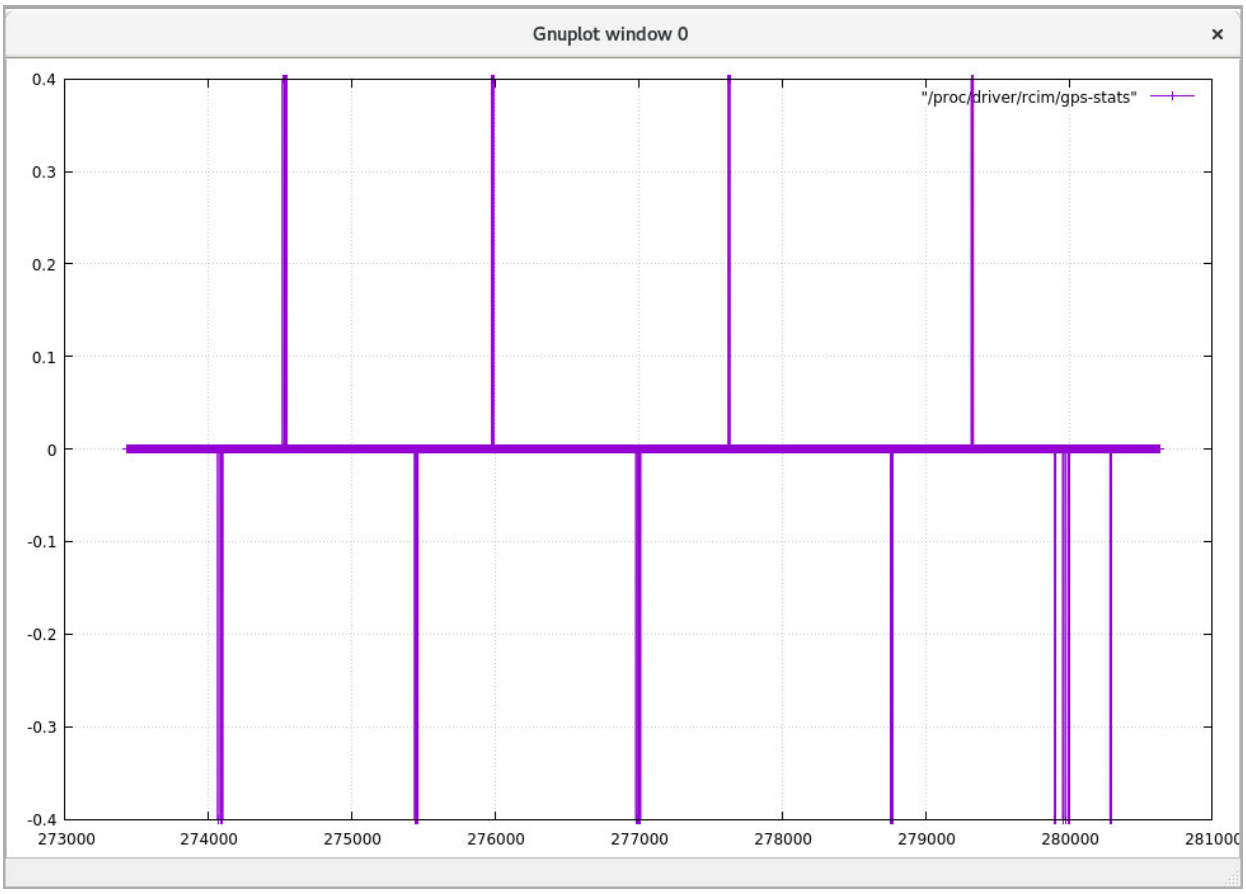
The **`gps-stats`** file can be viewed directly but is ideally suited for use with **`gnuplot`**. To use **`gnuplot`** to graph the statistics, first create a text file (e.g., named "`gps`") with the following contents:

```
set grid;
while (1) {
    plot "/proc/driver/rcim/gps-stats" with linespoints;
    pause 1;
}
```

and invoke **gnuplot** as follows:

```
gnuplot gps
```

The following screenshot shows an example of the **gnuplot** display, where the y-axis is the measured time difference in microseconds and the x-axis is the count of seconds since the system was booted.



See **gnuplot**(1) and invoke **info gnuplot** for more information.

5
IRIG Timecode Synchronization

This chapter describes how to configure an RCIM IV to be an IRIG master or slave.

Overview

The optional IRIG module allows an RCIM IV to function as an IRIG master or slave that is compliant with the IRIG Timecode specification. Both the IRIG-B B124 (AM) and the IRIG-B B004 (DCLS) formats are supported, along with a range of signal voltages and amplitudes.

RCIM IRIG masters can send IRIG signals to IRIG slaves, which can be either RCIM IRIG slaves or 3rd party IRIG slave cards.

RCIM IRIG slaves use **chronyd** to synchronize the RCIM's POSIX clock (the basis for system time) with an IRIG master. The capability to discipline system time on RCIM IRIG slaves is a key differentiator of the RCIM IRIG solution over 3rd party IRIG cards.

IRIG Master

RCIM IVs with IRIG modules default to being IRIG masters on startup, and RCIM IRIG masters automatically synchronize the IRIG time to the RCIM's POSIX clock with no manual configuration required.

RCIM IRIG master systems with disciplined system time (via GPS or a network timeserver) automatically synchronize the system time with the RCIM's POSIX clock, and this enables the IRIG time to automatically account for leap days, leap seconds and all other system time changes. See Chapter 4, "GPS Clock Synchronization" and **chronyd**(1) for more information.

Output Signals

RCIM IRIG masters can be configured to output analog Amplitude Modulation (AM) signals, digital DC Level Shift (DCLS) signals, or a combination of both. Details and configuration of the desired IRIG signal types are described in the following sections.

AM Signals

The RCIM IRIG master's AM signal output can output a range of voltages and should be compatible with most IRIG slaves. The detailed AM signal output specifications are:

- 2.5V, 5V or 8V peak-to-peak (selectable)
- 50 ohm impedance
- 1KHz at 3:1 ratio
- Accuracy of +/-10 microseconds

RCIM IRIG masters are automatically configured to output AM signals via external pin EXT_AMOUT. See Figure 3, "RCIM IV External Interrupt I/O Connector Pin-outs" for more information.

The AM signal output voltage defaults to 2.5V. If necessary, the AM signal output voltage can be manually set to either 2.5V, 5V or 8V using the output source configuration syntax. For example, to set the AM signal output voltage to 5V, invoke the following command as the root user:

```
echo '5V|irig_amp' > /proc/driver/rcim/config
```

Once you have decided which AM signal output voltage to use, you can add a line to **/etc/sysconfig/rcim** to cause the voltage to be configured automatically at boot time. For example:

```
RCIMCONFIG_3="5V|irig_amp"
```

See the comments in the **/etc/sysconfig/rcim** file for more information.

Note that RCIM IRIG masters can output only one AM signal. Refer to the next section if you wish to output one or more additional DCLS signals along with the AM signal.

DCLS Signals

The RCIM IRIG master's DCLS signal output supports multiple TTL voltages and should be compatible with most IRIG slaves. The detailed DCLS signal output specifications are:

- Compatible with 3.5V and 5V TTL with high >2V and low <0.5V
- 100 ohm impedance
- Accuracy of +/-1 microsecond

RCIM IRIG masters can output as many as twelve DCLS signals via any combination of the external pins EXT_INOUT0 through EXT_INOUT11. See Figure 3, "RCIM IV External Interrupt I/O Connector Pin-outs" for more information.

To enable an external pin to output a DCLS signal, configure the pin using output source configuration syntax. For example, to configure external pin EXT_INOUT2 to output a DCLS signal, invoke the following command as the root user:

```
echo 'dcls_out|out2,pin2/out/n' > /proc/driver/rcim/config
```

where:

`dcls_out`	selects the DCLS signal source
`out2`	specifies the source is an output
`pin2`	specifies which external pin to use (EXT_INOUT2)
`/out/n`	configures the pin to be an output with no termination

Once you have decided which external pin or pins will output DCLS signals, you can add lines to **`/etc/sysconfig/rcim`** to cause the pins to be configured automatically at boot time. For example:

```
RCIMCONFIG_4="dcls_out|out1,pin1/out/n"
RCIMCONFIG_5="dcls_out|out7,pin7/out/n"
```

See the comments in the **`/etc/sysconfig/rcim`** file for more information.

Propagation Delays

Advanced users can configure a timing offset to prematurely output IRIG master signals in order to compensate for signal propagation delays. The timing offset can be set to any value between 0 nanoseconds to 65,520 nanoseconds (65.52 microseconds) in 400 nanosecond intervals. By default the timing offset is 0 (no adjustment).

For example, to set the IRIG signal propagation to output 50 microseconds early, invoke the following command as the root user:

```
echo '50000|irig_adj' > /proc/driver/rcim/config
```

Once you have decided which IRIG signal propagation delay to use, you can add a line to **`/etc/sysconfig/rcim`** to cause the propagation delay to be configured automatically at boot time. For example:

```
RCIMCONFIG_6="50000|irig_adj"
```

See the comments in the **`/etc/sysconfig/rcim`** file for more information.

IRIG Slave

RCIM IVs with IRIG modules can be configured as IRIG slaves that receive IRIG signals from an IRIG master. By default, RCIMs are configured as IRIG masters, but running **`chronyd`** to synchronize the RCIM's POSIX clock with the IRIG PPS signal will automatically configure the RCIM to be an IRIG slave.

Because the RCIM's POSIX clock is the basis for system time, RCIM IRIG slave applications can quickly get precise IRIG time simply by calling **`clock_gettime`**(2). Calling **`clock_gettime`** will be much faster than reading multiple IRIG registers over the PCI Express bus using ioctls.

Configuration

Configure the RCIM IRIG slave by adding the following two `refclock` lines to **`/etc/chrony.conf`**:

```
refclock PPS /dev/pps1 lock PTP refid IRIG
refclock PHC /dev/ptp4 refid PTP noselect
```

NOTE

The example above uses **`/dev/pps1`** and **`/dev/ptp4`**, however the devices may be different on systems with different hardware configurations. To determine the correct devices for your system, search the kernel messages shortly after boot. For example:

```
# dmesg | grep -m1 -e 'pps.*ptp.*'
[35.205135] pps pps1: new PPS source ptp4
```

The above output indicates that **`/dev/pps1`** and **`/dev/ptp4`** are the correct devices for this system.

NOTE

The above two `refclock` lines must be the only `refclock` lines that exist in the **`/etc/chrony.conf`** file, ignoring any instances in commented out lines.

NOTE

The system must not run any other NTP or PTP daemons that synchronize the system time using a network-based time source. Ensure that any installed **`ntpd`** and **`ptpd`** services are disabled and stopped.

Once **`/etc/chrony.conf`** is configured correctly, issue the following commands as the root user to start **`chronyd`**:

```
systemctl enable chronyd
systemctl restart chronyd
```

Alternatively, issue the following commands on older systems without **`systemd`**:

```
chkconfig chronyd on
service chronyd start
```

Once IRIG PPS synchronization begins, the RCIM will be configured as an IRIG slave.

A major benefit of running **`chronyd`** on RCIM IRIG slaves is that it allows applications to simply call **`clock_gettime`**(2) to get the IRIG time, because system time is tightly synchronized to the IRIG time. This feature is unique to the RCIM IRIG slave solution,

however developers that wish to interface with the RCIM IRIG using device ioctls can refer to the "IRIG Programming Interface" section on page 5-8 for more information.

Input Signals

RCIM IRIG slaves can receive analog Amplitude Modulation (AM) signals or digital DC Level Shift (DCLS) signals. Each RCIM IRIG slave can receive only one IRIG signal input, either AM or DCLS, via an external pin. Details and configuration of the desired IRIG signal types are described in the following sections.

AM Signals

The RCIM IRIG slave's AM signal input is enabled by default. The AM signal input will accept a range of voltages and should be compatible with most IRIG masters. The detailed AM signal input specifications are:

- 1V to 10V peak-to-peak (auto ranging)
- 1KHz 2:1 to 6:1 ratio
- 4K ohm load AC coupled

To use the AM signal input, connect the IRIG master's AM signal output (EXT_AMOUT for IRIG RCIM masters) to the RCIM IRIG slave's AM signal input (EXT_AMIN). No additional configuration is required, other than running **`chronyd`** as described above.

To verify the RCIM IRIG slave is correctly synchronizing with the IRIG master, invoke the following command:

```
grep Slave /proc/driver/rcim/irig
```

Output similar to the following should be displayed:

```
IRIG Slave Input:    enabled: yes, PPS: good,
                     input type: AM
```

For additional verification, refer to the "Synchronization Accuracy" section on page 5-6.

DCLS Signals

The RCIM IRIG slave's DCLS signal input will accept a range of voltages and should be compatible with most IRIG masters. The detailed DCLS signal input specifications are:

- 3.3V or 5V TTL with high >2V and low <0.8V
- 100 ohm switchable termination

To use a DCLS signal input, connect the IRIG master's DCLS signal output to an external pin on the RCIM IRIG slave and configure the pin using the input source configuration syntax. For example, to configure external pin EXT_INOUT7 to input a DCLS signal, invoke the following command as the root user:

```
echo 'pin7|dcls_in,pin7/in/n' > /proc/driver/rcim/config
```

where:

`pin7`	specifies the source is an input
`dcls_in`	selects the DCLS signal source
`pin7`	specifies which external pin to use (EXT_INOUT7)
`/in/n`	configures the pin to be an input with no termination

NOTE

Once DCLS signal input is enabled, AM signal input is disabled. AM signal input can be re-enabled by rebooting or by issuing the following command:

```
echo 'none|dcls_in' > /proc/driver/rcim/config
```

To verify the RCIM IRIG slave is correctly synchronizing with the IRIG master, invoke the following command:

```
grep Slave /proc/driver/rcim/irig
```

Output similar to the following should be displayed:

```
IRIG Slave Input:    enabled: yes, PPS: good,
                     input type: DCLS, input pin: 7
```

For additional verification steps, refer to "Synchronization Accuracy" on page 5-6.

Once you have decided which external pin will receive DCLS signals, you can add lines to **`/etc/sysconfig/rcim`** to cause the pins to be configured automatically at boot time. For example:

```
RCIMCONFIG="pin5|dcls_in,pin5/in/n"
```

See the comments in the **`/etc/sysconfig/rcim`** file for more information.

Synchronization Accuracy

During synchronized IRIG slave operation, the RCIM IRIG slave's POSIX clock (the basis for system time) should be synchronized to within one microsecond of the IRIG PPS signal when using DCLS input signals, and to within ten microseconds of the IRIG PPS signal when using AM input signals.

When synchronization is occurring, synchronization accuracy is measured once per second and the values for the last two hours of operation are temporarily recorded in the **`/proc/driver/rcim/irig-stats`** file.

NOTE

On RedHawk 7.5 and earlier only one hour of history is recorded.

The accuracy measured is the difference in nanoseconds between the RCIM POSIX clock (the basis for system time) and the IRIG PPS that is driven directly by signals from the IRIG master.

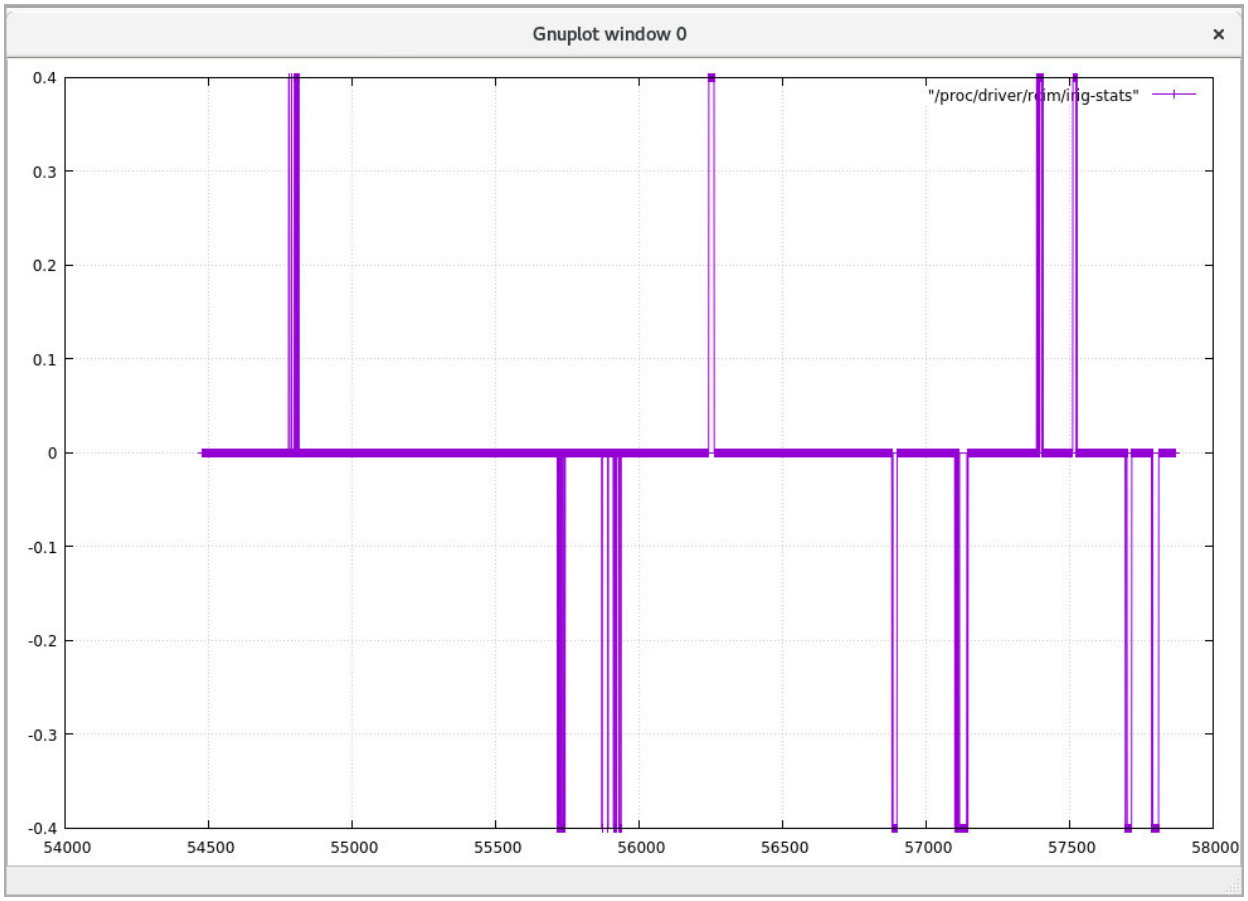
The **irig-stats** file can be viewed directly but is ideally suited for use with **gnuplot**. To use **gnuplot** to graph the statistics, first create a text file (e.g., named "irig") with the following contents:

```
set grid;
while (1) {
    plot "/proc/driver/rcim/irig-stats" with linespoints;
    pause 1;
}
```

and invoke **gnuplot** as follows:

```
gnuplot irig
```

The following screenshot shows an example of the **gnuplot** display, where the y-axis is the measured time difference in microseconds and the x-axis is the count of seconds since the system was booted:



See **gnuplot**(1) and invoke **info gnuplot** for more information.

IRIG Programming Interface

NOTE

Use of the IRIG programming interface is generally unnecessary. IRIG masters and slaves can be configured, accessed, and controlled without any programming required, as described in the "IRIG Master" section on page 5-1 and the "IRIG Slave" section on page 5-3 above.

The RCIM driver provides several ioctls that can be used to access the IRIG registers and state information that are present when the RCIM card has an IRIG module. To use the provided ioctls, first open **/dev/rcim:0/irig** to get a valid file descriptor for the RCIM IRIG module, and then invoke the desired ioctls with the file descriptor.

By default, **/dev/rcim:0/irig** has file permissions that allow any user to read from and write to the device, however a udev rule can be used to limit access. For example, on RedHawk 8.x, placing the following into **/etc/udev/rules.d/99-irig.rules** will limit device access to the root user and other users that belong to a new `irig` group:

```
KERNEL=="rcim:0/irig", OWNER:="root", GROUP:="irig", MODE:="0660"
```

Upon reboot, the **/dev/rcim:0/irig** device file will have these permissions:

```
# ls -l /dev/rcim:0/irig
crw-rw---- 1 root irig 508, 224 Aug  8 15:24 /dev/rcim:0/irig
```

Two sets of ioctls are provided: one set for accessing information on IRIG masters, and another set for accessing information on IRIG slaves. Details for each set of ioctls are provided in the following sections. See **ioctl**(2) for more information.

Advanced users can also **mmap**(2) the **/dev/rcim:0/irig** device to access the RCIM IRIG registers directly. See Appendix A, "RCIM IV Registers" for more information.

IRIG Master Ioctls

The following sections describe the ioctls that are provided for accessing IRIG registers and state information on RCIM IRIG masters. Any arguments shown are additional ioctl arguments that appear after the initial file descriptor argument for **/dev/rcim:0/irig**.

IRIG_OUTPUT_ENABLE

Synopsis: Enable IRIG master operation.

Description: The RCIM will begin to continuously output the RCIM's POSIX clock in IRIG timecode format.

Arguments: None.

IRIG_OUTPUT_DISABLE

Synopsis: Disable IRIG master operation.

Description: The RCIM will stop outputting the RCIM's POSIX clock time in IRIG timecode format.

Arguments: None.

IRIG_OUTPUT_STATUS

Synopsis: Query IRIG master current state.

Description: Copy the current state into the integer pointed to by the argument. A status of `1` indicates the RCIM is outputting IRIG timecode signals. A status of `0` indicates the RCIM is not outputting IRIG timecode signals.

Arguments: `int *`

IRIG_OUTPUT_RESET

Synopsis: Reset IRIG master.

Description: Clear all IRIG master programmed information (e.g., output type, output pin).

Arguments: None.

IRIG_OUTPUT_GET_CONTROL_BITS

Synopsis: Query IRIG Output Control Bits Register.

Description: The IRIG Output Control Bits Register holds information for inclusion in the IRIG timecode signals that are output by IRIG masters and may contain user-defined fields. Copy this register into the integer pointed to by the argument. Only the lower 18 bits of the value are significant.

Arguments: `int *`

IRIG_OUTPUT_SET_CONTROL_BITS

Synopsis: Set IRIG Output Control Bits Register.

Description: The IRIG Output Control Bits Register holds information for inclusion in the IRIG timecode signals that are output by IRIG masters and may contain user-defined fields. Copy the integer pointed to by the argument into this register. Only the lower 18 bits of the value are significant.

Arguments: `int *`

IRIG_GET_LEAP_SECOND

Synopsis: Query IRIG master leap second.

Description: Query IRIG master programmed information for time of next leap second insertion and write the time into the structure pointed to by the argument. The time written is in `irig_master_leap_second_args` format, which is included with **`rcim.h`** and defined as follows:

```
struct irig_master_leap_second_args {
   unsigned int year;
   unsigned int day;
   unsigned int hour;
}
```

Arguments: `struct irig_master_leap_second_args *`

IRIG_SET_LEAP_SECOND

Synopsis: Set IRIG master leap second.

Description: Programs the IRIG master to perform a leap second insertion at the 59th second of the 59th minute of the year, day, and hour specified by the time structure pointed to by the argument. Year, day and hour are represented using `irig_master_leap_second_args` format, which is included with **`rcim.h`** and defined as follows:

```
struct irig_master_leap_second_args {
   unsigned int year;
   unsigned int day;
   unsigned int hour;
}
```

Arguments: `struct irig_master_leap_second_args *`

IRIG Slave Ioctls

Using ioctls to access the RCIM IRIG information requires reads and writes of the RCIM IRIG registers over the PCI Express bus. For RCIM IRIG slaves, using these ioctls to access IRIG time will be significantly slower than calling **`clock_gettime`**. See **`clock_gettime`**(2) for more information.

The following sections describe the ioctls that are provided for accessing IRIG registers and state information on RCIM IRIG slaves. Any arguments shown are additional ioctl arguments that appear after the initial file descriptor argument for **`/dev/rcim:0/irig`**.

IRIG_INPUT_ENABLE

Synopsis: Enable IRIG slave operation.

Description: When enabled, if a valid AM or DCLS signal is present, then the RCIM will provide an IRIG PPS signal for **`chronyd`** to use to synchronize the RCIM's POSIX clock.

Arguments: None.

IRIG_INPUT_DISABLE

Synopsis: Disable IRIG slave operation.

Description: When disabled, the RCIM will no longer provide an IRIG PPS signal and enter a reset state. All IRIG slave programmed information (e.g., input type, input pin) will be cleared.

Arguments: None.

IRIG_INPUT_STATUS

Synopsis: Query IRIG slave current state.

Description: Copy the current state into the integer pointed to by the argument. A status of `1` indicates IRIG slave operation is currently enabled. A status of `0` indicates IRIG slave operation is currently disabled.

Arguments: `int *`

IRIG_GET_TIME

Synopsis: Query IRIG slave current time.

Description: Copy the IRIG slave's latest time snapshot into the structure pointed to by the argument using `rtc_time` format, which is included with **`rtc.h`** and defined as follows:

NOTE

Only the `tm_sec`, `tm_min`, `tm_hour`, `tm_year` and `tm_yday` structure fields are updated.

```
struct rtc_time {
   int tm_sec;
   int tm_min;
   int tm_hour;
   int tm_mday;
   int tm_mon;
```

```
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
}
```

Arguments: `struct rtc_time *`

IRIG_GET_NS64

Synopsis: Query IRIG slave current time in nanoseconds.

Description: Copy the IRIG slave's latest time snapshot in nanoseconds into the value pointed to by the argument. The snapshot is nanoseconds since the epoch (midnight UTC on January 1, 1970) using `ns64` format, which is included with **`rcim.h`** and defined as a typedef of a signed 64-bit integer.

Arguments: `ns64 *`

IRIG_GET_TIME_RAW

Synopsis: Query IRIG slave current raw time.

Description: Copy the IRIG slave's latest raw time snapshot into the structure pointed to by the argument using `rtc_time` format, which is included with **`rtc.h`** and defined below:

NOTE

Only the `tm_sec`, `tm_min`, `tm_hour`, `tm_year` and `tm_yday` structure fields are updated. In addition, the raw time snapshot holds only the two least significant digits for `tm_year` and has a `tm_sec` value that is always one second in the past.

```
struct rtc_time {
    int tm_sec;
    int tm_min;
    int tm_hour;
    int tm_mday;
    int tm_mon;
    int tm_year;
    int tm_wday;
    int tm_yday;
    int tm_isdst;
}
```

Arguments: `struct rtc_time *`

A
RCIM IV Registers

This section contains the address map and registers on the RCIM IV board.

Note that some registers appear at two places in the physical address space. For these registers there is an associated *Xregister*. For example PCSAT and XPCSAT. The *Xregister* accommodates systems with a 64k, rather than the old 4k page size.

RCIM IV Address Map

Address	Function	
0xXXX00000	Board Status/Control Register	
0xXXX00004	Firmware Rev/Options Present Register	(Read Only)
0xXXX00008	Board Identity Register	
0xXXX00010	Interrupt Enable Register #1	
0xXXX00014	Interrupt Enable Register #2	
0xXXX00020	Interrupt Request Register #1	(Write Only)
0xXXX00024	Interrupt Request Register #2	(Write Only)
0xXXX00020	Interrupt Pending Register #1	(Read Only)
0xXXX00024	Interrupt Pending Register #2	(Read Only)
0xXXX00028	Interrupt Pending Extra Register #1	(Read Only)
0xXXX0002C	Interrupt Pending Extra Register #2	(Read Only)
0xXXX00030	Interrupt Clear Register #1	
0xXXX00034	Interrupt Clear Register #2	
0xXXX00040	Interrupt Arm Register #1	
0xXXX00044	Interrupt Arm Register #2	
0xXXX00050	Interrupt Select Level Register #1	
0xXXX00054	Interrupt Select Level Register #2	
0xXXX00060	Interrupt Select Polarity Register #1	
0xXXX00064	Interrupt Select Polarity Register #2	
0xXXX00070	External Interrupt Routing Register #1	
0xXXX00074	External Interrupt Routing Register #2	
0xXXX00078	External Interrupt Routing Register #3	
0xXXX00080	Cable Interrupt Routing Register #1	
0xXXX00084	Cable Interrupt Routing Register #2	
0xXXX00088	Cable Interrupt Routing Register #3	
0xXXX00200	GPS PPS Snapshot Register	(Read Only)
0xXXX00204	GPS PPS Seconds Snapshot Register	(Read Only)
0xXXX00210	Cable Snapshot Register	(Read Only)
0xXXX00214	Cable Seconds Snapshot Register	(Read Only)
0xXXX00220	Cable Master Time Register	(Read Only)
0xXXX00230	IRIG PPS Snapshot Register	(Read Only)
0xXXX00234	IRIG PPS Seconds Snapshot Register	(Read Only)

Address	Function	
0xXXX00400	Clear Cable Errors Register	(Write Only)
0xXXX00410	Output Cable Status Register	(Read Only)
0xXXX00420	Input Cable Status Register	(Read Only)
0xXXX01000 0xXXX10000	Tick Clock Upper Register Tick Clock Upper Register	
0xXXX01008 0xXXX10008	Tick Clock Lower Register Tick Clock Lower Register	
0xXXX01010 0xXXX10010	Tick Clock Status/Control Register Tick Clock Status/Control Register	
0xXXX01100 0xXXX10100	POSIX Clock Seconds Register POSIX Clock Seconds Register	
0xXXX01108 0xXXX10108	POSIX Clock Nanoseconds Register POSIX Clock Nanoseconds Register	
0xXXX01110 0xXXX10110	POSIX Clock Status/Control Register POSIX Clock Status/Control Register	
0xXXX01114 0xXXX10114	POSIX Clock Skip/Add Time Register POSIX Clock Skip/Add Time Register	(Write Only) (Write Only)
0xXXX01120 0xXXX10120	Clock Frequency Adjust Register Clock Frequency Adjust Register	
0xXXX01150	External Clock Input Select	
0xXXX01500- 0xXXX015F0	MSI-X Lower Address N	
0xXXX01504- 0xXXX015F4	MSI-X Upper Address N	
0xXXX01508- 0xXXX015F8	MSI-X Message Data N	
0xXXX0150C- 0xXXX015FC	MSI-X Vector Control Register N	
0xXXX01600	MSI-X Pending Register	(Read Only)
0xXXX01700- 0xXXX01778	MSI-X Select #1 Register N	
0xXXX01704- 0xXXX0177C	MSI-X Select #2 Register N	
0xXXX01800- 0xXXX01878	MSI-X Time #1 Register N	
0xXXX01804- 0xXXX0187C	MSI-X Time #2 Register N	
0xXXX01900- 0xXXX0193C	MSI-X Interrupt Clear Register #1 N	(Write Only)
0xXXX02000	RTC #0 Control Register	
0xXXX02010	RTC #0 Timer Register	
0xXXX02014	RTC #0 Repeat Register	
0xXXX02020	RTC #1 Control Register	
0xXXX02030	RTC #1 Timer Register	
0xXXX02034	RTC #1 Repeat Register	
0xXXX02040	RTC #2 Control Register	
0xXXX02050	RTC #2 Timer Register	
0xXXX02054	RTC #2 Repeat Register	

Address	Function	
0xXXX02060	RTC #3 Control Register	
0xXXX02070	RTC #3 Timer Register	
0xXXX02074	RTC #3 Repeat Register	
0xXXX02080	RTC #4 Control Register	
0xXXX02090	RTC #4 Timer Register	
0xXXX02094	RTC #4 Repeat Register	
0xXXX020A0	RTC #5 Control Register	
0xXXX020B0	RTC #5 Timer Register	
0xXXX020B4	RTC #5 Repeat Register	
0xXXX020C0	RTC #6 Control Register	
0xXXX020D0	RTC #6 Timer Register	
0xXXX020D4	RTC #6 Repeat Register	
0xXXX020E0	RTC #7 Control Register	
0xXXX020F0	RTC #7 Timer Register	
0xXXX020F4	RTC #7 Repeat Register	
0xXXX03000 0xXXX30000	Programmable Interrupt Generator Register Programmable Interrupt Generator Register	
0xXXX03010 0xXXX30010	Programmable INTR Generator Set Register Programmable INTR Generator Set Register	(Write Only) (Write Only)
0xXXX03020 0xXXX30020	Programmable INTR Generator Clear Register Programmable INTR Generator Clear Register	(Write Only) (Write Only)
0xXXX03040	External I/O Output Enable Register	
0xXXX03044	External I/O Output Enable Set Register	
0xXXX03048	External I/O Output Enable Clear Register	
0xXXX03050	External I/O Terminator On Register	
0xXXX03054	External I/O Terminator On Set Register	
0xXXX03058	External I/O Terminator On Clear Register	
0xXXX03200	GPS Receive Pointers Register	
0xXXX03204	GPS Transmit Pointers Register	
0xXXX03208	GPS Debug Control/Status Register	
0xXXX0320C	GPS Communication Error Register	
0xXXX03300	Board Information Register	(Read Only)
0xXXX03304	Firmware Date Register	(Read Only)
0xXXX03308	Firmware Revision Register	(Read Only)
0xXXX03800	Firmware SPI Command/Status Register	
0xXXX03804	Firmware SPI Address Register	
0xXXX03808	Firmware Reload Register	
0xXXX0380C	Remote Update Sector/Status Register	
0xXXX03900- 0xXXX039FC	Firmware SPI Buffer	
0xXXX04000- 0xXXX047FF	GPS Receive Data Buffer	
0xXXX04800- 0xXXX04FFF	GPS Transmit Data Buffer	
0xXXX06000	IRIG Input Enable Register	
0xXXX06004	IRIG Input Control Register	

Address	Function	
0xXXX06008	IRIG Input Status Register	(Read Only)
0xXXX0600C	IRIG Input Error Register	(Read Only)
0xXXX06020	IRIG Input Seconds Register	(Read Only)
0xXXX06024	IRIG Input Minutes Register	(Read Only)
0xXXX06028	IRIG Input Hours Register	(Read Only)
0xXXX0602C	IRIG Input Days Register	(Read Only)
0xXXX06030	IRIG Input Years Register	(Read Only)
0xXXX06034	IRIG Input Control Bits Register	(Read Only)
0xXXX06038	IRIG Input SBS Register	(Read Only)
0xXXX06060	IRIG Output Enable Register	
0xXXX06064	IRIG Output Control Register	
0xXXX06068	IRIG Output Adjust Register	
0xXXX0606C	IRIG Output Leap Second Time Register	
0xXXX06080	IRIG Output Seconds Register	
0xXXX06084	IRIG Output Minutes Register	
0xXXX06088	IRIG Output Hours Register	
0xXXX0608C	IRIG Output Days Register	
0xXXX06090	IRIG Output Years Register	
0xXXX06094	IRIG Output Control Bits Register	
0xXXX06098	IRIG Output SBS Register	
0xXXX06100	IRIG ADC Data Register (Debug)	(Read Only)
0xXXX06110- 0xXXX0614C	IRIG ADC History Registers (Debug)	(Read Only)
0xXXX06160	IRIG DAC Data Register (Debug)	
0xXXX06164	IRIG DA C Command/Data Register (Debug)	

RCIM IV Registers

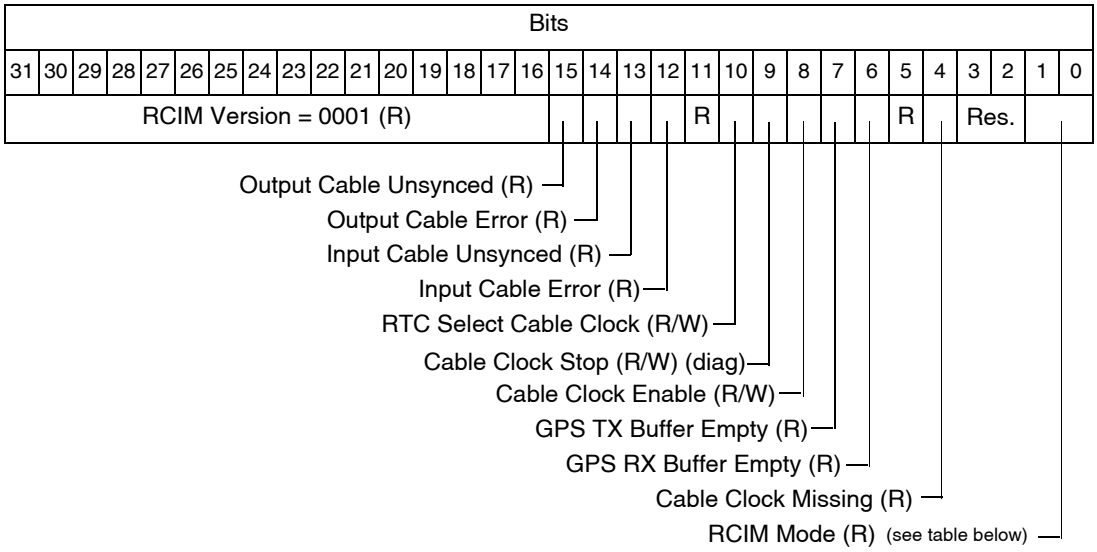
RCIM IV registers are illustrated in this section.

NOTE: Unless otherwise stated, a bit value of 1=on; 0=off

Figure A-1 RCIM IV Board Status/Control Register

This register provides status and control of certain features of the RCIM IV board.

Offset: 0x00000



RCIM Mode Bit Values

Bits 1 0	Description	Input Cable	Output Cable	Cable Clock
1 1	Stand-alone Master RCIM	Open	Open	Inactive
1 0	Master RCIM	Open	Connected	Sourcing
0 1	Last Slave RCIM	Connected	Open	Receiving
0 0	Pass-thru Slave RCIM	Connected	Connected	Receiving

Figure A-2 RCIM IV Firmware Revision/Options Present Register

This register provides information on what options are present on this RCIM board and the firmware revision.

Offset: 00004

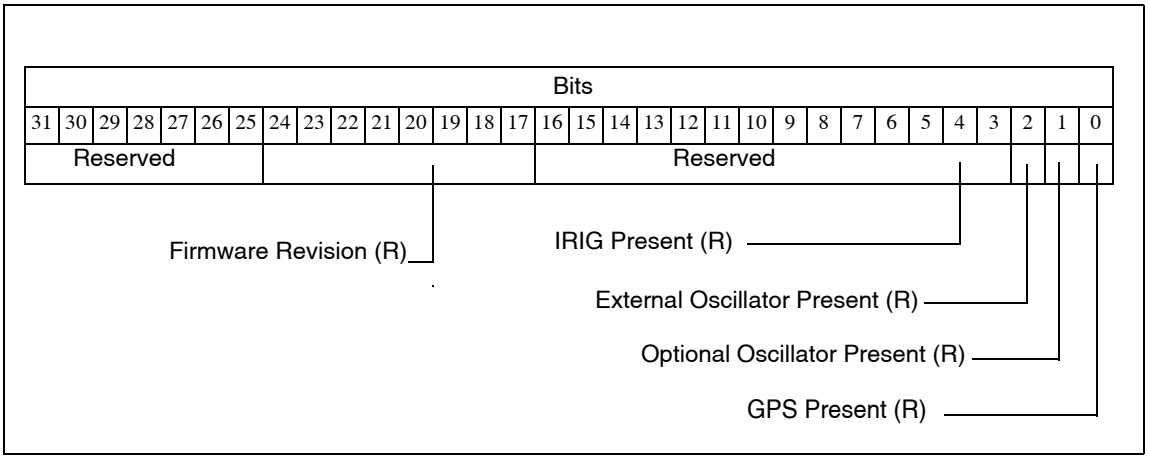


Figure A-3 RCIM IV Board Identity Register

Setting bit 7 causes the RCIM LEDs to flash red and green once per second to identify the specific board, useful when there are multiple RCIMs in a one system. Clearing bit 7 will resume normal operation.

Offset: 00008

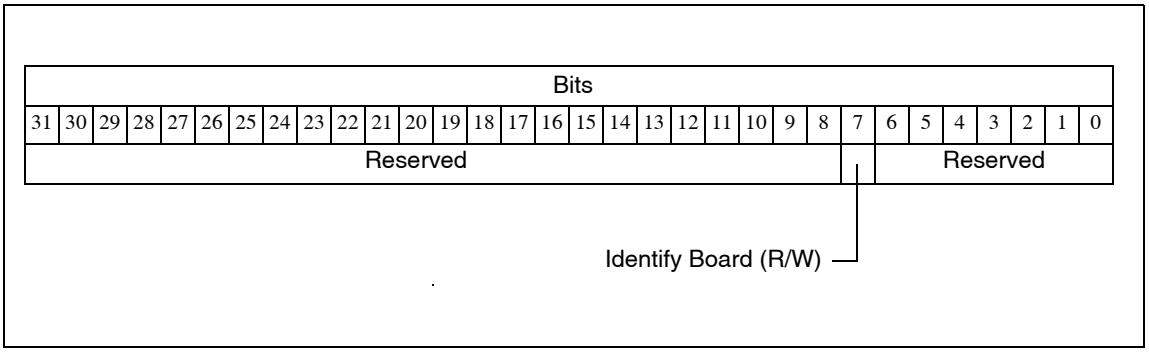


Figure A-4 RCIM IV Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers

The enable registers enable the selected interrupts.
The request registers are software driven requests of the selected interrupts.
The pending registers are pending requests.
The clear registers clear the selected interrupts.
The arm registers arm the selected interrupts for edge triggering.
The level registers set level (1) or edge (0) for the selected interrupts.
The polarity registers set polarity high (1) or low (0) for the selected interrupts.

Offsets: 00010, 00014, 00020, 00024, 00030, 00034, 00040, 00044, 00050, 00054, 00060, 00064

Register #1

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CI								AUX				Reserved				EI								RTC							
7	6	5	4	3	2	1	0	3	2	1	0					7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Register #2

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				CI				Reserved												EI				Reserved							
				11	10	9	8													11	10	9	8								

CI = Cable Interrupt
AUX = Auxiliary Interrupt
EI = External Interrupt
RTC = RTC Interrupt

AUX 0 = GPS Pulse Per Second
AUX 1 = GPS TX Buffer Empty
AUX 2 = GPS RX Buffer Full
AUX 3 = IRIG Pulse Per Second

NOTE

There is one exception to the above register map: bit 16 of the RCIM IV Interrupt Pending Register #1 at offset 00020 is not reserved. When bit 16 is read as "1" it indicates that there are no interrupts pending in the RCIM IV Interrupt Pending Register #2 at offset 00024.

Figure A-5 RCIM IV External Interrupt Routing Registers

The external interrupt routing registers route selected interrupts to the external interrupt connector.

Offset: 00070, 00074, 00078

External Interrupt Number — Register Number

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.		3						Res.		2						Res.		1						Res.		0					#1
		7								6								5								4					#2
		11								10								9								8					#3
		5	4	3	2	1	0			5	4	3	2	1	0			5	4	3	2	1	0			5	4	3	2	1	0

Value	Description
0x00	Reserved
0x01	Reserved
0x02	Reserved
0x03	Reserved
0x04	RTC 0
0x05	RTC 1
0x06	RTC 2
0x07	RTC 3
0x08	Edge Interrupt 0
0x09	Edge Interrupt 1
0x0A	Edge Interrupt 2
0x0B	Edge Interrupt 3
0x0C	PIG 0
0x0D	PIG 1
0x0E	PIG 2
0x0F	PIG 3
0x10	Cable Interrupt 0
0x11	Cable Interrupt 1
0x12	Cable Interrupt 2
0x13	Cable Interrupt 3
0x14	Cable Interrupt 4
0x15	Cable Interrupt 5
0x16	Cable Interrupt 6
0x17	Cable Interrupt 7
0x18	Cable Interrupt 8
0x19	Cable Interrupt 9
0x1A	Cable Interrupt 10
0x1B	Cable Interrupt 11
0x1C	Reserved
0x1D	Reserved
0x1E	Reserved
0x1F	Reserved

Value	Description
0x20	GPS PPS
0x21	IRIG PPS
0x22	Reserved
0x23	Reserved
0x24	RTC 4
0x25	RTC 5
0x26	RTC 6
0x27	RTC 7
0x28	Edge Interrupt 4
0x29	Edge Interrupt 5
0x2A	Edge Interrupt 6
0x2B	Edge Interrupt 7
0x2C	PIG 4
0x2D	PIG 5
0x2E	PIG 6
0x2F	PIG 7
0x30	IRIG DCLS
0x31	Reserved
0x32	Reserved
0x33	Reserved
0x34	10 MHz Clock
0x35	Master Time Signal
0x36	Reserved
0x37	Reserved
0x38	Edge Interrupt 8
0x39	Edge Interrupt 9
0x3A	Edge Interrupt 10
0x3B	Edge Interrupt 11
0x3C	PIG 8
0x3D	PIG 9
0x3E	PIG 10
0x3F	PIG 11

Figure A-6 RCIM IV Cable Interrupt Routing Registers

The cable interrupt routing registers route selected interrupts to the RCIM interconnecting cable.

Offsets: 00080, 00084, 00088

Cable Interrupt Register 1

Cable Interrupt Number

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7				6				5				4				3				2				1				0			
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Cable Interrupt Register 2

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																11				10				9				8			
																3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

Cable Interrupt Register 3

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								11		10		9		8		7		6		5		4		3		2		1		0	
								5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4

Value	Description	Value	Description	Value	Description
0x00	Reserved	0x10	Reserved	0x20	GPS PPS
0x01	Reserved	0x11	Reserved	0x21	IRIG PPS
0x02	Reserved	0x12	Reserved	|	|
0x03	Reserved	0x13	Reserved	0x27	Reserved
0x04	RTC 0	0x14	RTC 4	0x28	Edge Interrupt 8
0x05	RTC 1	0x15	RTC 5	0x29	Edge Interrupt 9
0x06	RTC 2	0x16	RTC 6	0x2A	Edge Interrupt 10
0x07	RTC 3	0x17	RTC 7	0x2B	Edge Interrupt 11
0x08	Edge Interrupt 0	0x18	Edge Interrupt 4	0x2C	PIG 8
0x09	Edge Interrupt 1	0x19	Edge Interrupt 5	0x2D	PIG 9
0x0A	Edge Interrupt 2	0x1A	Edge Interrupt 6	0x2E	PIG 10
0x0B	Edge Interrupt 3	0x1B	Edge Interrupt 7	0x2F	PIG 11
0x0C	PIG 0	0x1C	PIG 4	0x30	Reserved
0x0D	PIG 1	0x1D	PIG 5	|	|
0x0E	PIG 2	0x1E	PIG 6	|	|
0x0F	PIG 3	0x1F	PIG 7	0x3F	Reserved

Figure A-7 RCIM IV GPS PPS Snapshot Register

The GPS PPS Snapshot register contains a snapshot of the nanoseconds field and two bits of the seconds field of the POSIX clock. The snapshot is taken every time the GPS PPS signal occurs.

Offset: 00200

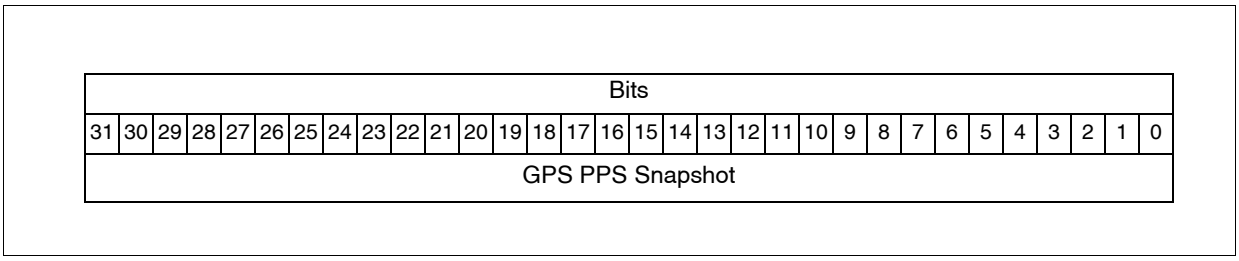


Figure A-8 RCIM IV GPS PPS Seconds Snapshot Register

The GPS PPS Seconds Snapshot register contains a snapshot of the seconds field of the POSIX clock. The snapshot is taken every time the GPS PPS signal occurs.

Offset: 00204

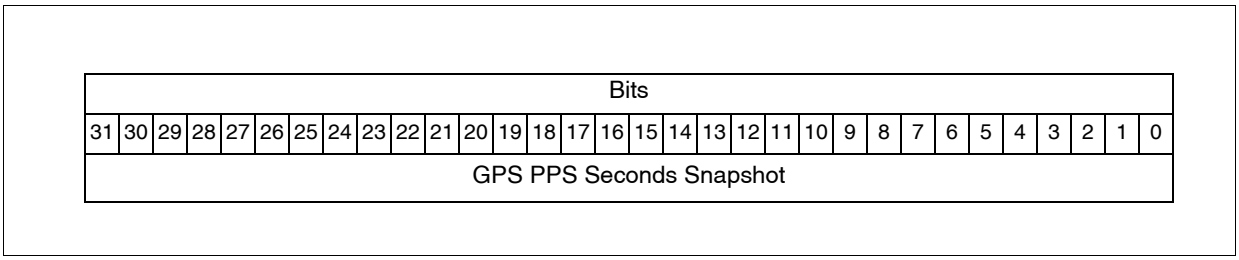


Figure A-9 RCIM IV Cable Snapshot Register

The Cable Snapshot register contains a snapshot of the nanoseconds field and two bits of the seconds field of the POSIX clock. The snapshot is taken every time the cable master time is received.

Offset: 00210

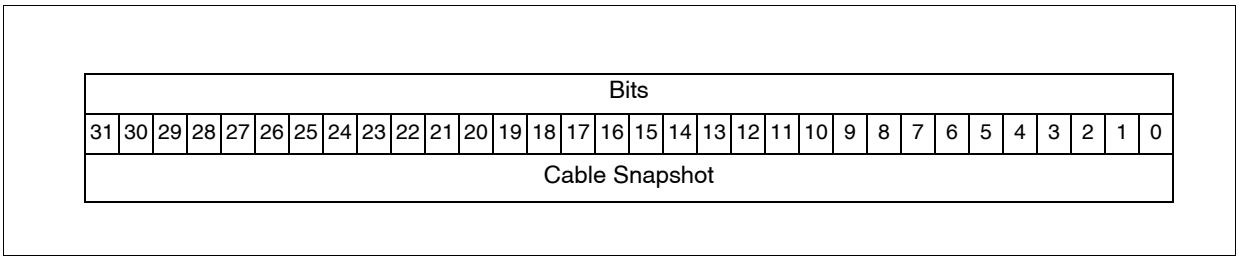


Figure A-10 RCIM IV Cable Seconds Snapshot Register

The Cable Seconds Snapshot register contains a snapshot of the seconds field of the POSIX clock. The snapshot is taken every time the cable master time is received.

Offset: 00214

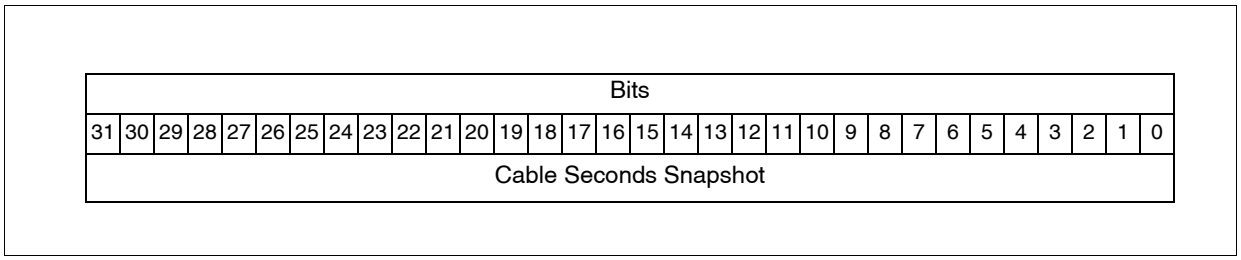


Figure A-11 RCIM IV IRIG PPS Snapshot Register

The IRIG PPS Snapshot register contains a snapshot of the nanoseconds field and two bits of the seconds field of the POSIX clock. The snapshot is taken every time the IRIG PPS signal occurs.

Offset: 00230

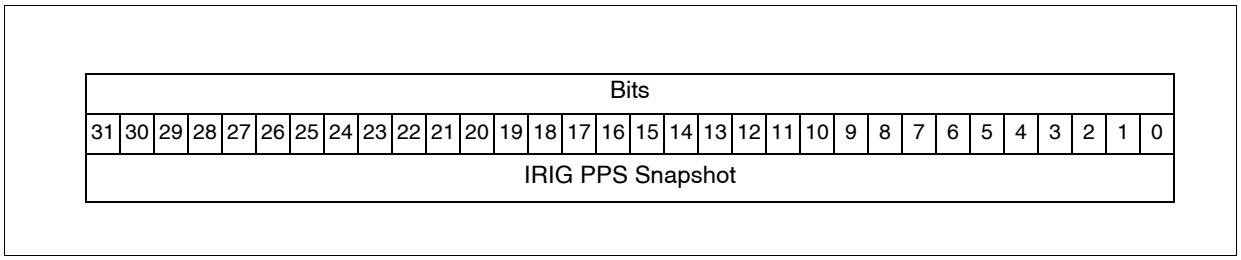


Figure A-12 RCIM IV IRIG PPS Seconds Snapshot Register

The IRIG PPS Seconds Snapshot register contains a snapshot of the seconds field of the POSIX clock. The snapshot is taken every time the IRIG PPS signal occurs.

Offset: 00234

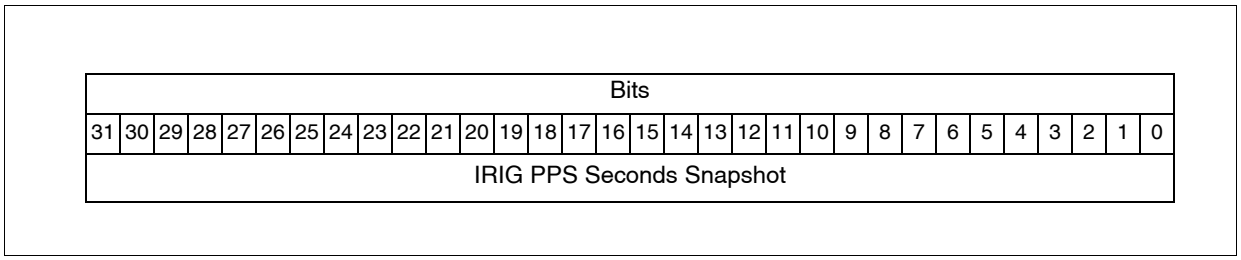


Figure A-13 RCIM IV Cable Master Time Register

The Cable Master Time register contains the seconds field of the master RCIM POSIX clock that is transmitted on the cable at every transition of the clock at the seconds boundary.

Offset: 00220

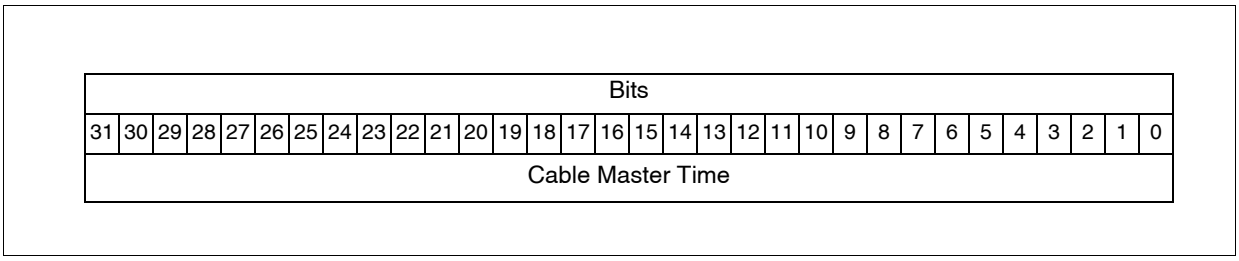


Figure A-14 RCIM IV Clear Cable Errors Register

This is a Write Only register that clears any reported cable errors. The data field is don't care.

Offset: 00400

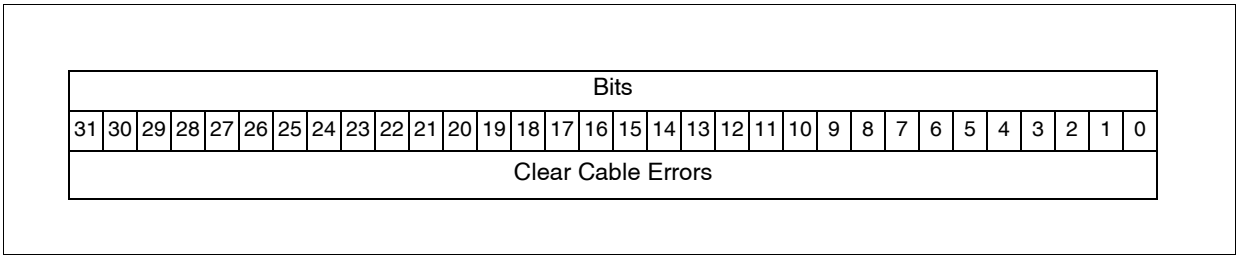


Figure A-15 RCIM IV Output Cable Status Register

This register provides detailed hardware status information pertaining to the output cable.

Offset: 00410

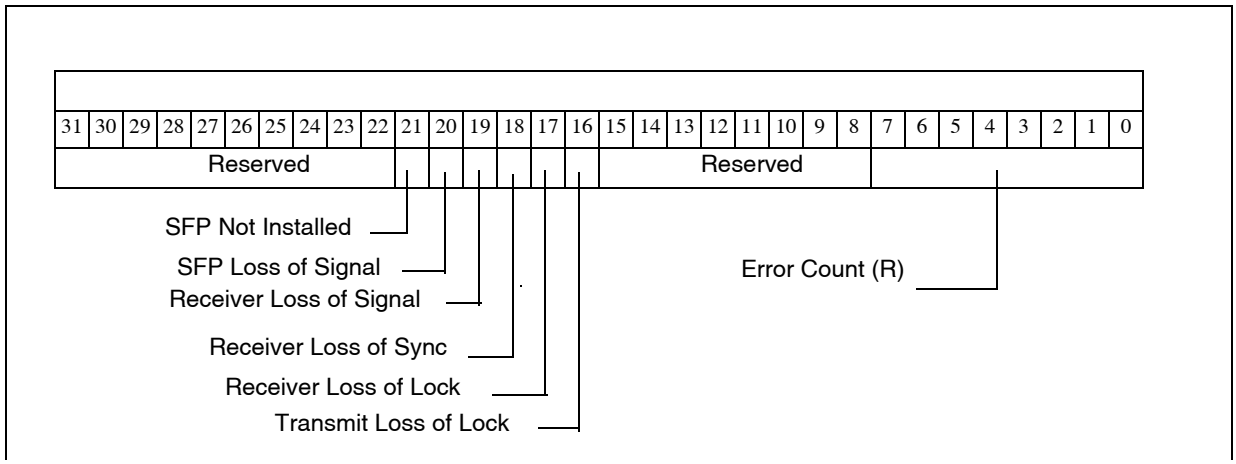


Figure A-16 RCIM IV Input Cable Status Register

This register provides detailed hardware status information pertaining to the input cable.

Offset: 00420

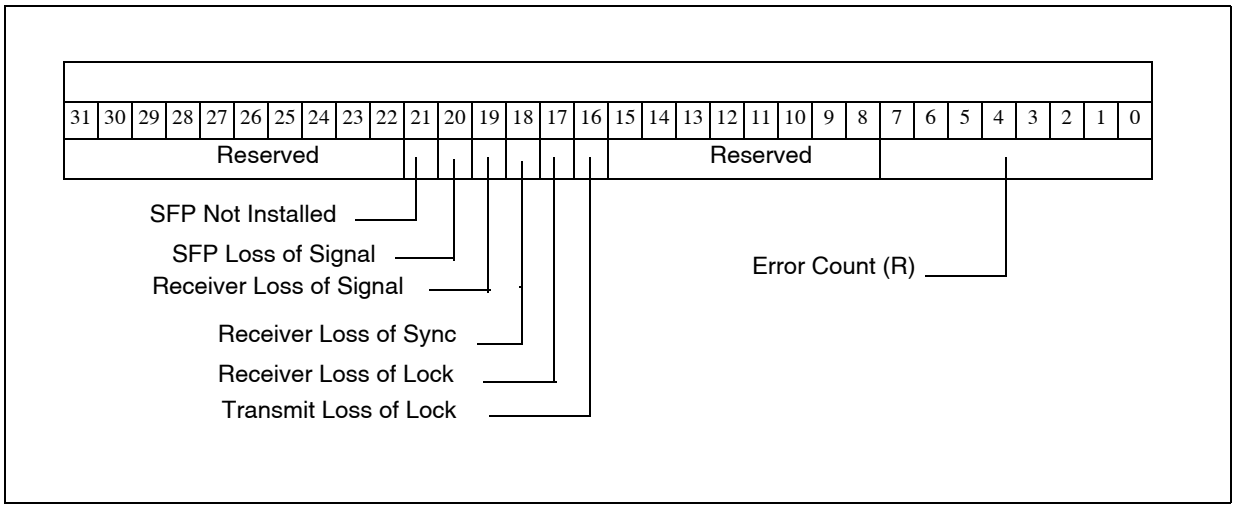


Figure A-17 RCIM IV Tick Clock Upper Register

This register contains the upper 32 bits of the tick clock.

Offsets: 01000, 10000

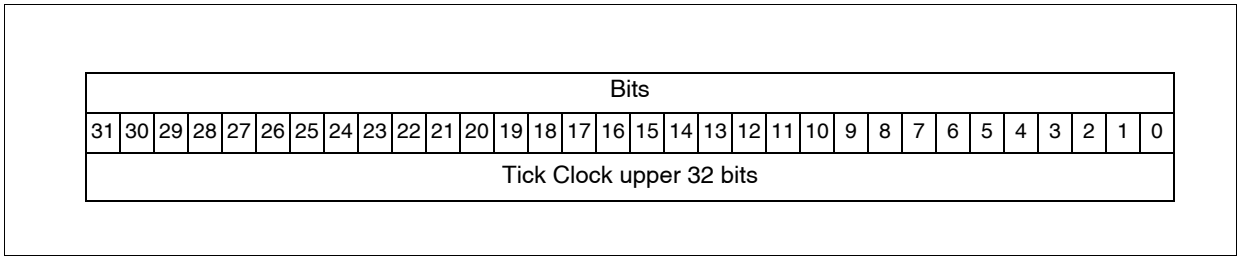


Figure A-18 RCIM IV Tick Clock Lower Register

This register contains the lower 32 bits of the tick clock.

Offsets: 01008, 10008

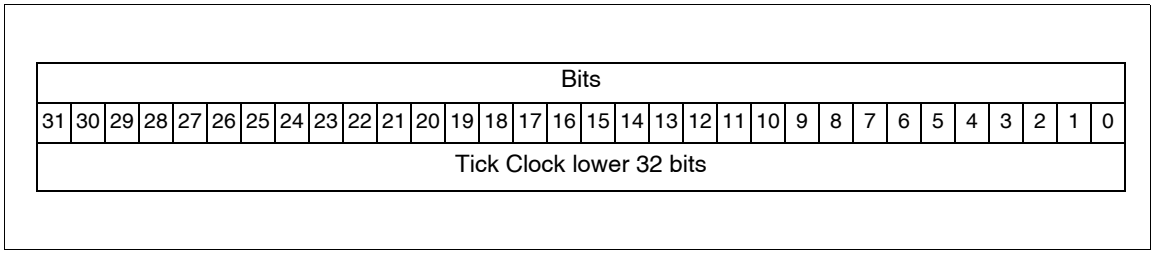


Figure A-19 RCIM IV Tick Clock Status/Control Register

This register provides status and control of the tick clock.

Offsets: 01010, 10010

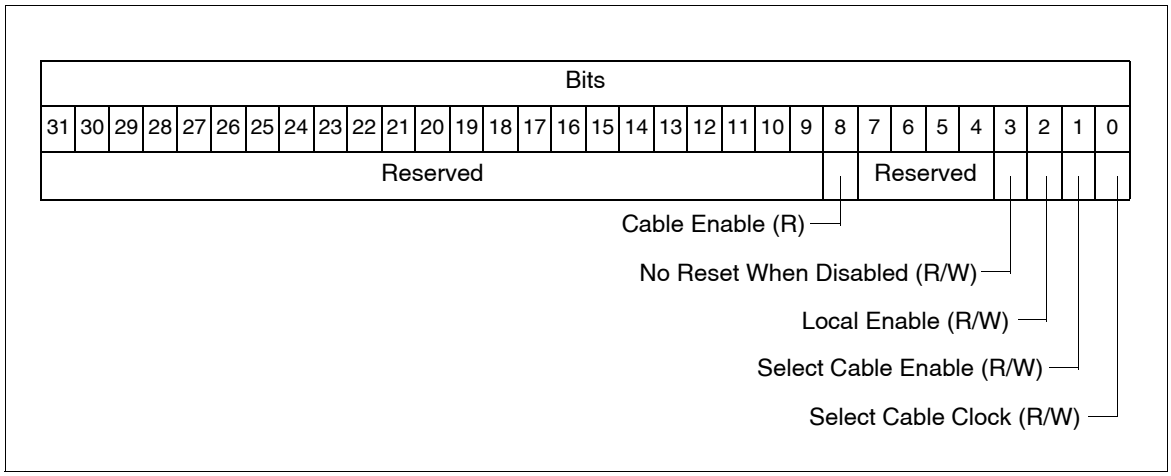


Figure A-20 RCIM IV POSIX Clock Seconds Register

This register contains the POSIX clock seconds.

Offsets: 01100, 10100

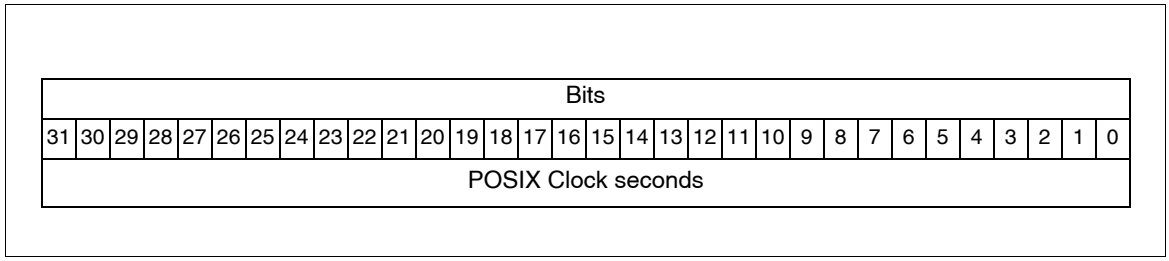


Figure A-21 RCIM IV POSIX Clock Nanoseconds Register

This register contains the POSIX clock nanoseconds.

Offsets: 01108, 10108

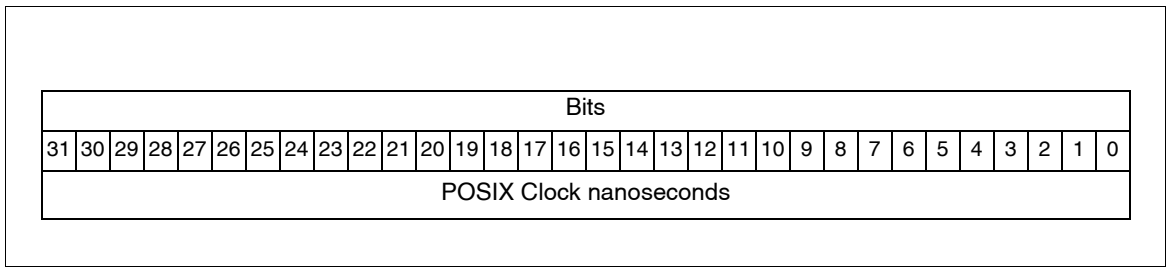


Figure A-22 RCIM IV POSIX Clock Status/Control Register

This register provides status and control of the POSIX clock.

Offsets: 01110, 10110

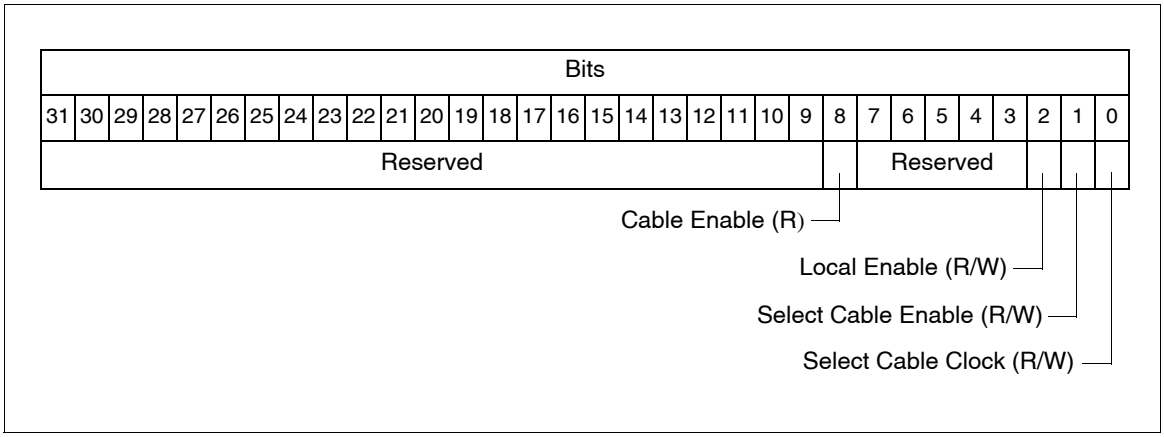


Figure A-23 RCIM IV POSIX Clock Skip/Add Time Register

This register skips/adds time to the POSIX clock in 400 nanosecond increments.

Offsets: 01114, 10114

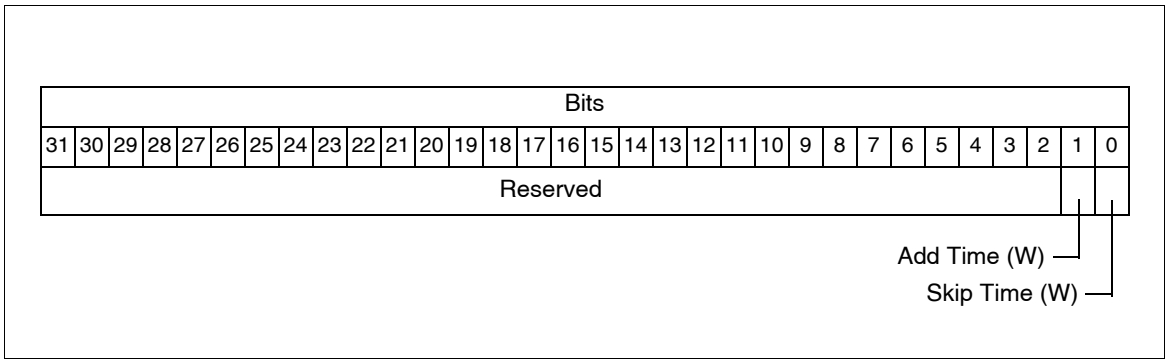


Figure A-24 RCIM IV Clock Frequency Adjust Register

The clock frequency adjust register is used to control the frequency of the 10 MHz master clock.

Offsets: 01120, 10120

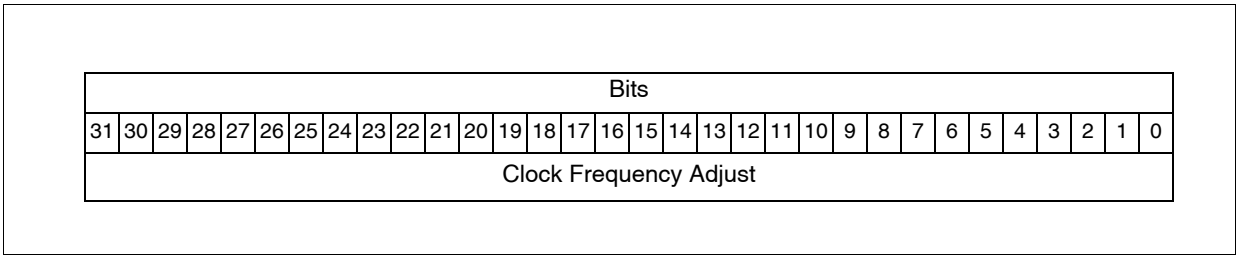
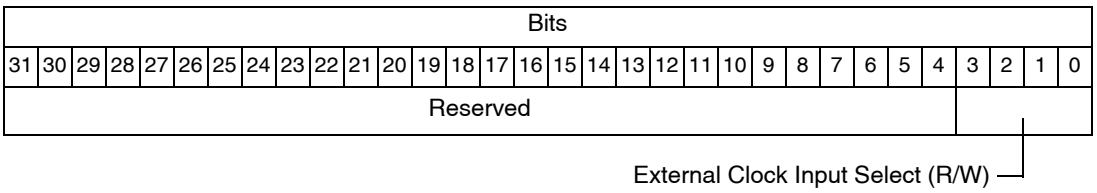


Figure A-25 RCIM IV External Clock Input Select Register

This register selects the external clock input source for a 10 MHz signal to drive the RCIM clock.

Offsets: 01150



External Clock Input Select

Value	Input Source
0	External input 0
1	External input 1
2	External input 2
3	External input 3
4	External input 4
5	External input 5
6	External input 6
7	External input 7
8	External input 8
9	External input 9
10	External input 10
11	External input 11
12-14	Reserved
15	None (Default)

Figure A-26 RCIM IV MSI-X Lower Address N Registers

The lower 32-bits of the message address for each MSI-X interrupt.

Offsets: 1500, 1510, 1520, 1530, 1540, 1550, 1560, 1570, 1580, 1590, 15A0, 15B0, 15C0, 15D0, 15E0, 15F0

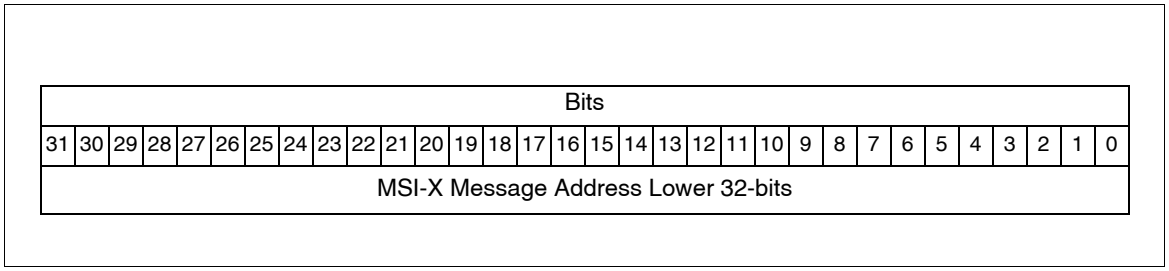


Figure A-27 RCIM IV MSI-X Upper Address N Registers

The upper 32-bits of the message address for each MSI-X interrupt.

Offsets: 1504, 1514, 1524, 1534, 1544, 1554, 1564, 1574, 1584, 1594, 15A4, 15B4, 15C4, 15D4, 15E4, 15F4

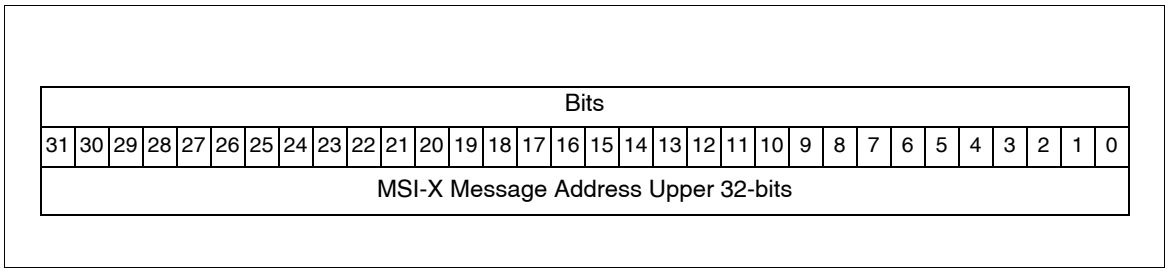


Figure A-28 RCIM IV MSI-X Message Data N Registers

The message data for each MSI-X interrupt.

Offsets: 1508, 1518, 1528, 1538, 1548, 1558, 1568, 1578, 1588, 1598, 15A8, 15B8, 15C8, 15D8, 15E8, 15F8

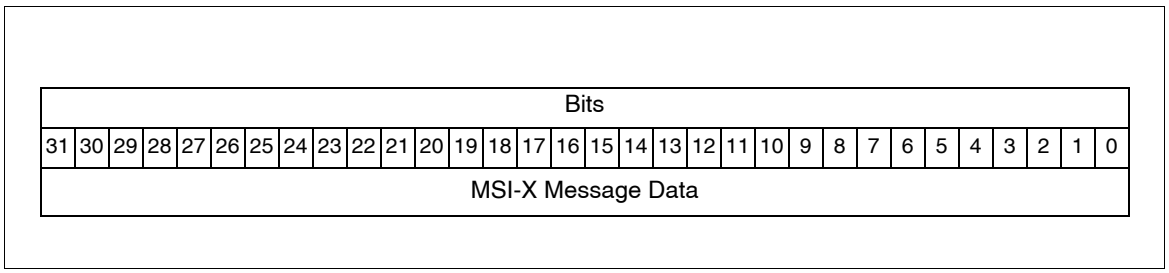


Figure A-29 RCIM IV MSI-X Vector Control N Registers

The vector control registers for each MSI-X interrupt.

Offsets: 150C, 151C, 152C, 153C, 154C, 155C, 156C, 157C, 158C, 159C, 15AC, 15BC, 15CC, 15DC, 15EC, 15FC

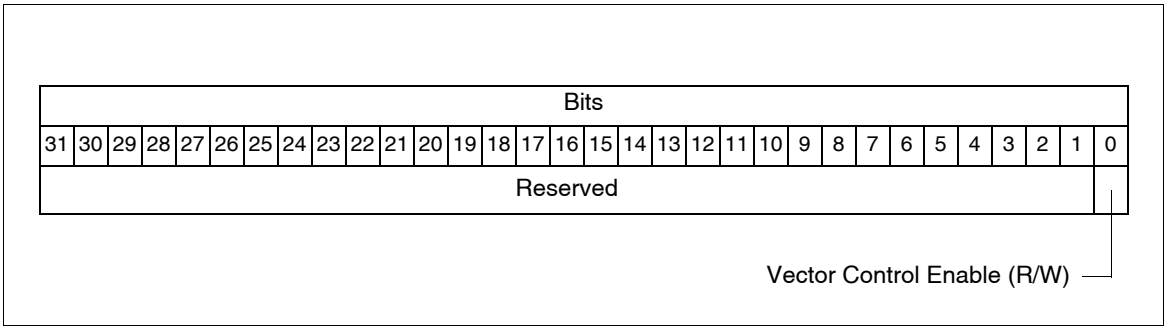


Figure A-30 RCIM IV MSI-X Pending Registers

The MSI-X pending register indicates the set of MSI-X interrupts that are currently pending. These bits will not be visible very long if the interrupts are enabled.

Offsets: 1600

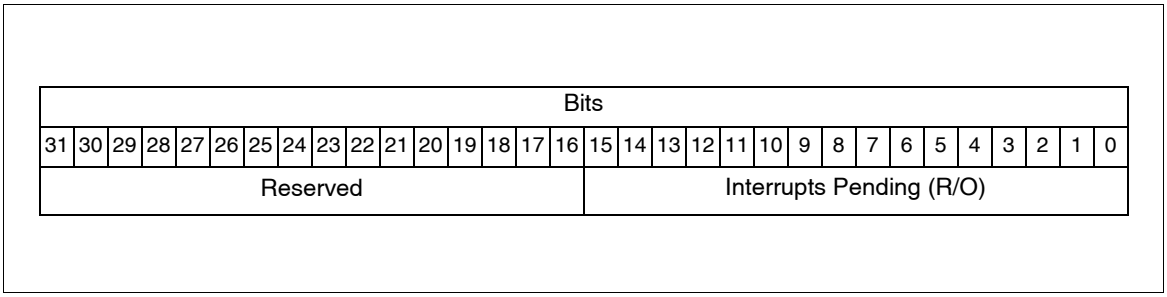


Figure A-31 RCIM IV MSI-X Select #1 Register N

These registers are used to select the individual interrupts for each MSI-X interrupt.

Offsets: 1700, 1708, 1710, 1718, 1720, 1728, 1730, 1738, 1740, 1748, 1750, 1758, 1760, 1768, 1770, 1778

These registers have the same mapping as the Interrupt Enable Register #1. Refer to Figure A-4, "RCIM IV Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers," on page A-7.

Figure A-32 RCIM IV MSI-X Select #2 Register N

These registers are used to select the individual interrupts for each MSI-X interrupt.

Offsets: 1704, 170C, 1714, 171C, 1724, 172C, 1734, 173C, 1744, 174C, 1754, 175C, 1764, 176C, 1774, 177C

These registers have the same mapping as the Interrupt Enable Register #2. Refer to Figure A-4, “RCIM IV Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers,” on page A-7.

Figure A-33 RCIM IV MSI-X Time #1 Register N

These registers contains snapshots of the nanoseconds field, along with the lower two bits of the seconds field, of the POSIX clock taken every time each MSI-X interrupt occurs.

Offsets: 1800, 1808, 1810, 1818, 1820, 1828, 1830, 1838, 1840, 1848, 1850, 1858, 1860, 1868, 1870, 1878

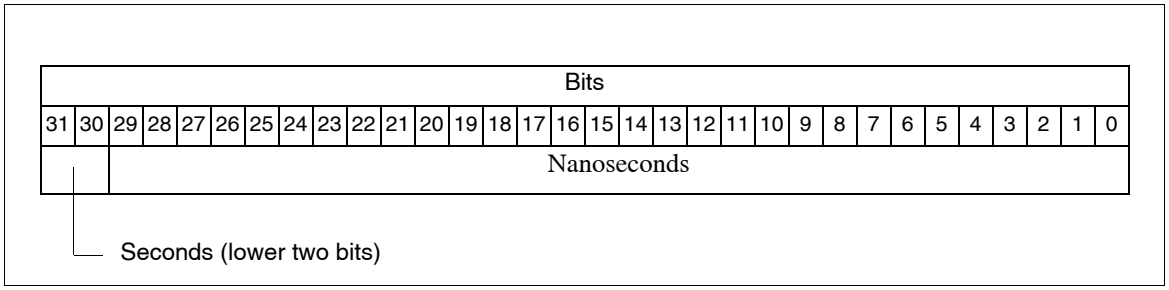


Figure A-34 RCIM IV MSI-X Time #2 Register N

These registers contain snapshots of the seconds field of the POSIX clock taken every time each MSI-X interrupt occurs.

Offsets: 1804, 180C, 1814, 181C, 1824, 182C, 1834, 183C, 1844, 184C, 1854, 185C, 1864, 186C, 1874, 187C

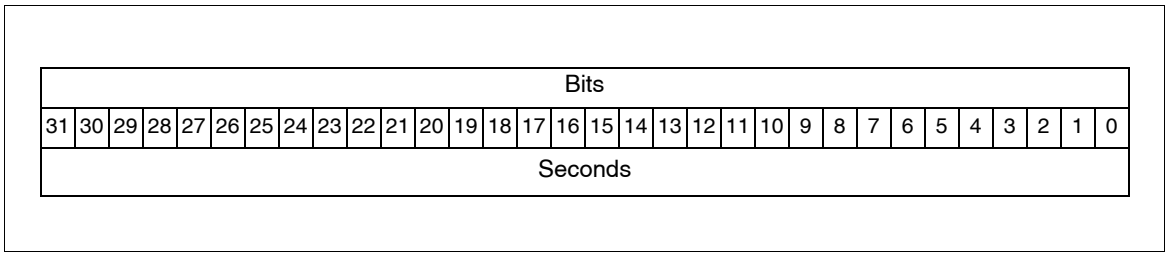


Figure A-35 RCIM IV MSI-X Interrupt Clear Register #1 N

These registers clear each MSI-X interrupt.

Offsets: 1900, 1904, 1908, 190C, 1910, 1914, 1918, 191C, 1920, 1924, 1928, 192C, 1930, 1934, 1938, 193C

These registers have the same mapping as the Interrupt Clear Register #1. Refer to Figure A-4, “RCIM IV Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers,” on page A-7.

This is a write only register. Writes to the register will clear the selected interrupt.

Note: This is the same interrupt clear register used for all interrupts. The address mapping for the individual MSI-X interrupts is used to allow new interrupts to be generated without the use of enable/disable bit writes. Any writes to the Interrupt Clear Register #2 for an MSI-X interrupt MUST be followed by a write to this register even if there are no interrupts selected in this range.

Figure A-36 RCIM IV RTC Timer Registers

The initial RTC timer value is loaded in the RTC timer registers. The current value of the timer is read from this register. NOTE: Loading this register also loads the RTC Repeat Register for compatibility with RCIM.

Offsets: 02010, 02030, 02050, 02070, 02090, 020B0, 020D0, 020F0

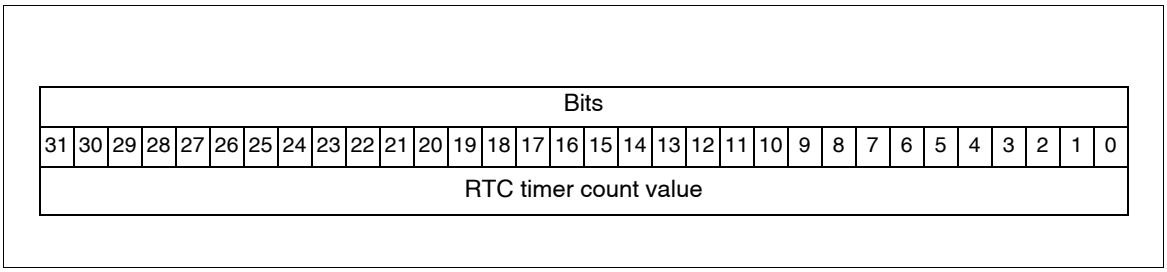


Figure A-37 RCIM IV RTC Repeat Registers

The RTC repeat registers contain the repeat count value.

Offsets: 02014, 02034, 02054, 02074, 02094, 020B4, 020D4, 020F4

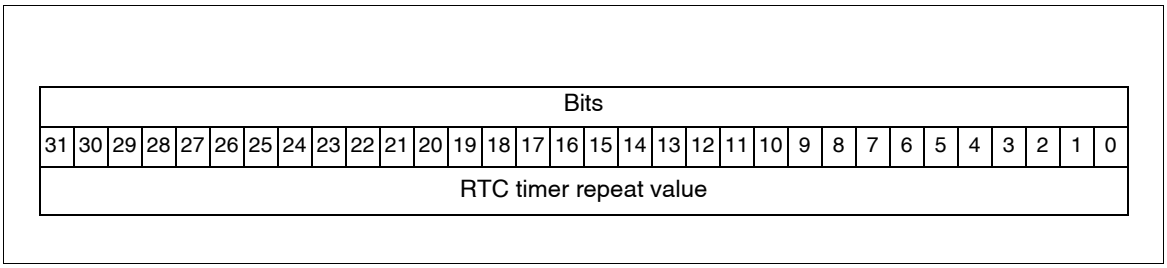
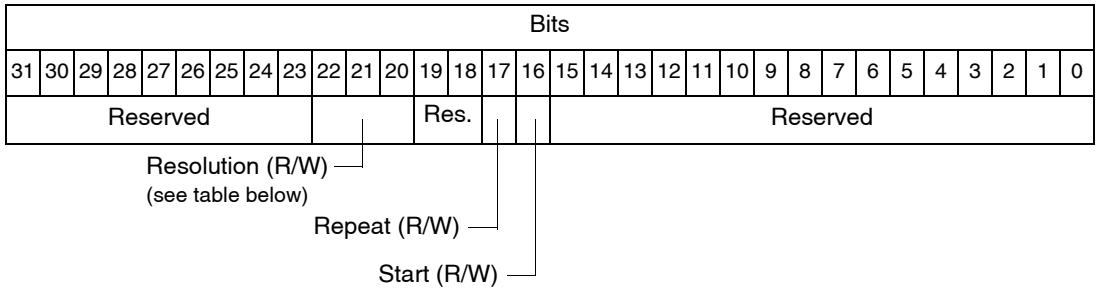


Figure A-38 RCIM IV RTC Control Registers

This register provides control of the RTCs.

Offsets: 02000, 02020, 02040, 02060, 02080, 020A0, 020C0, 020E0



Resolution Bit Values

Bits 22	21	20	Description
0	0	0	1 microsecond
0	0	1	10 microseconds
0	1	0	100 microseconds
0	1	1	1 millisecond
1	0	0	10 milliseconds
1	0	1	stopped
1	1	0	stopped
1	1	1	stopped

Figure A-39 RCIM IV Programmable Interrupt Generator Register

This register identifies the programmable interrupts.

Offsets: 03000, 30000

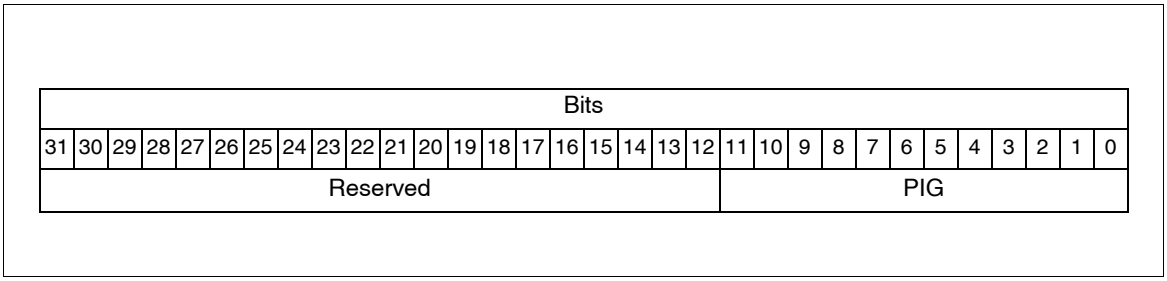


Figure A-40 RCIM IV Programmable Interrupt Set and Clear Registers

Writing to these registers sets/clears the unitary bits in the Programmable Interrupt Register without affecting the other bits.

Offsets: 03010, 30010 03020, 30020

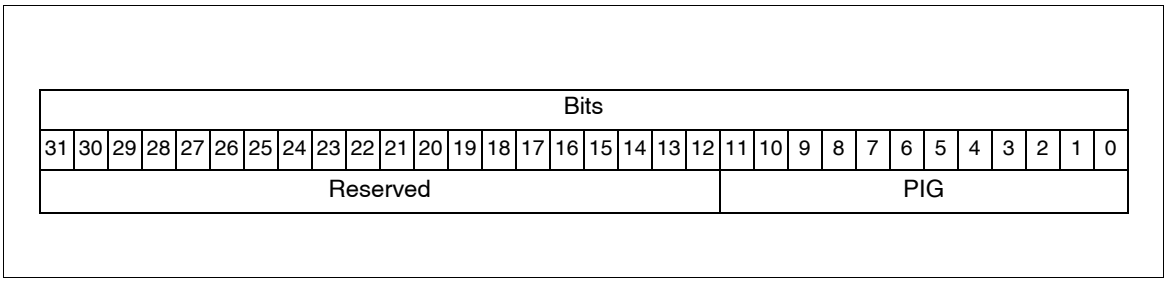


Figure A-41 RCIM IV External I/O Output Enable Register

This register sets or resets the selected external I/O output enable.

Offsets: 03040

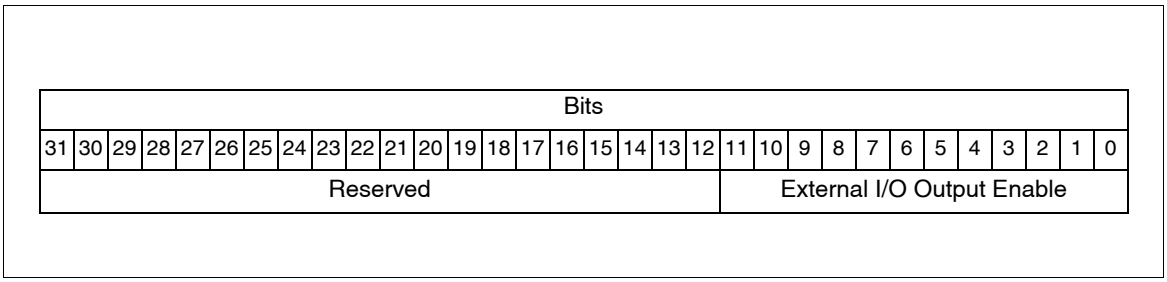


Figure A-42 RCIM IV External I/O Output Enable Set/Clear Registers

Writing to these registers sets/clears the unitary bits in the external I/O output enable register without affecting the other bits.

Offsets: 03044, 03048

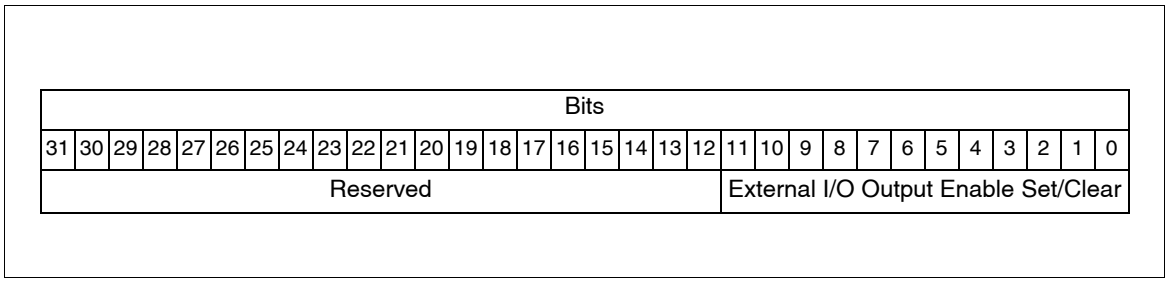


Figure A-43 RCIM IV External I/O Terminator On Register

This register sets or resets the selected external I/O terminator on.

Offsets: 03050

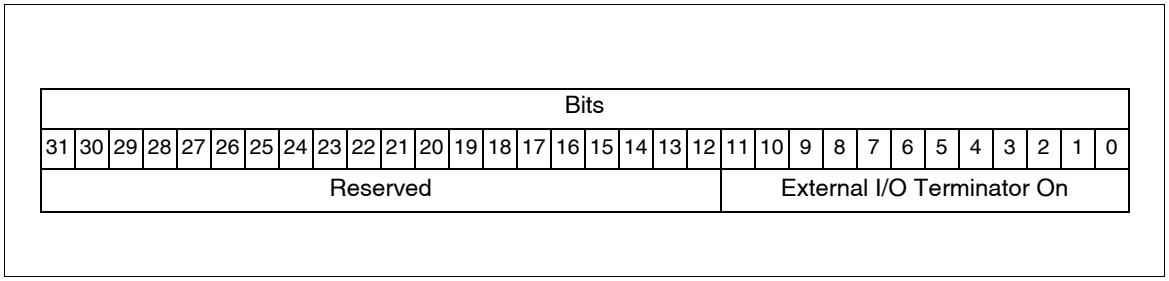


Figure A-44 RCIM IV External I/O Terminator On Set/Clear Registers

Writing to these registers sets/clears the unitary bits in the external I/O terminator on register without affecting the other bits.

Offsets: 03054, 03058

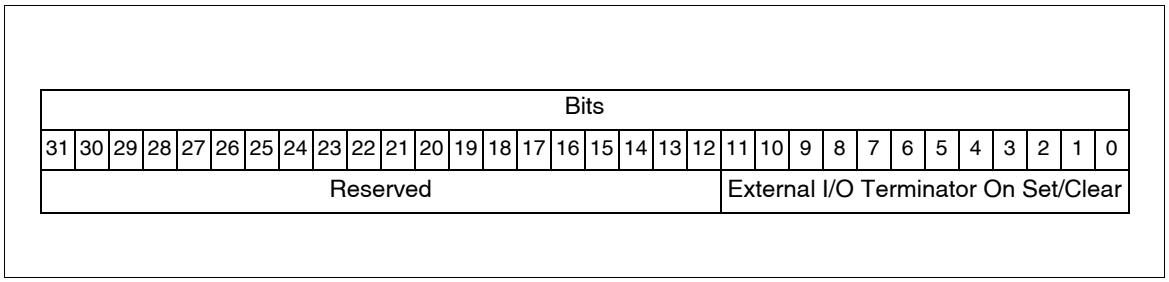


Figure A-45 RCIM IV GPS Receive Pointers

The GPS receive pointers are used for communication with the optional GPS module.

Offset: 03200

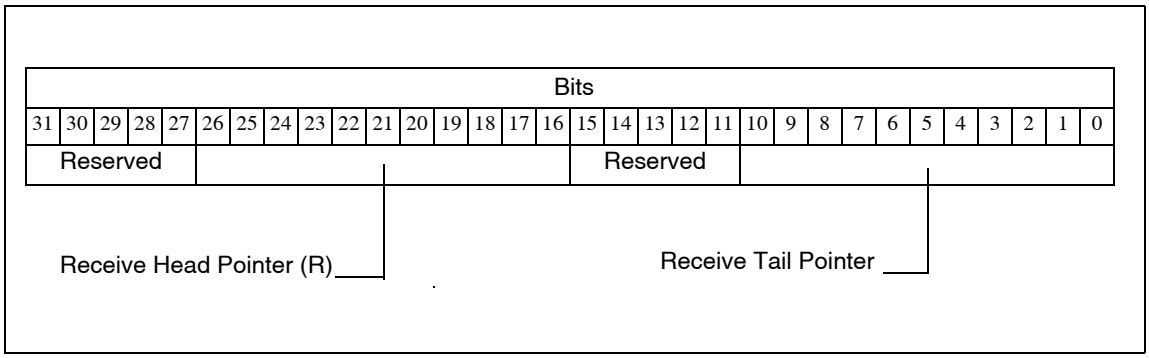


Figure A-46 RCIM IV GPS Transmit Pointers

The GPS transmit pointers are used for communication with the optional GPS module.

Offset: 03204

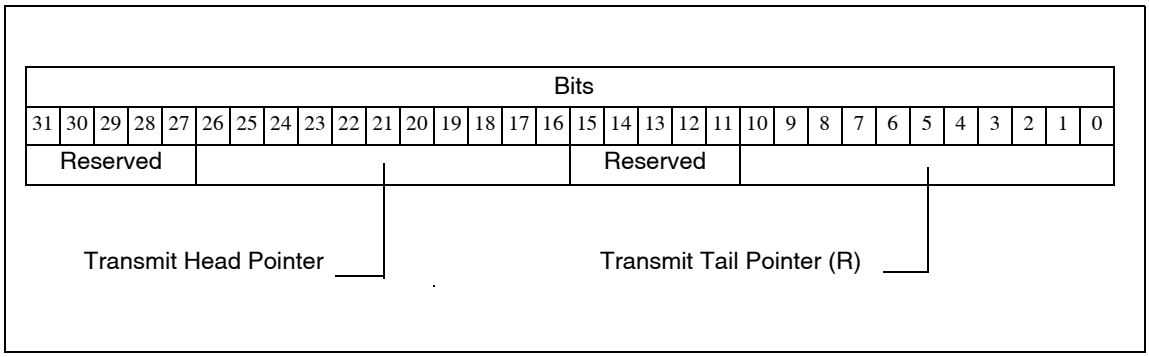


Figure A-47 RCIM IV GPS Debug Control/Status Register

The GPS debug control/status register contains bits used during testing and debug. Setting any of these bits will disable RCIM communication with the GPS module.

Offset: 03208

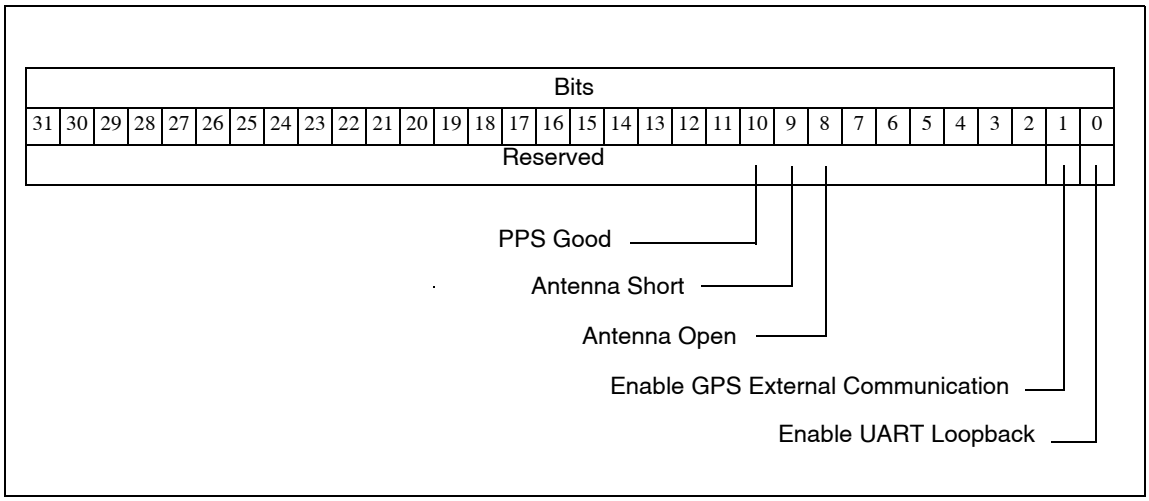


Figure A-48 RCIM IV GPS Communication Error Register

The GPS communication error register contains information regarding communication errors with the GPS module.

Any write to this register will reset the communication interface to the GPS module.

Offset: 0320C

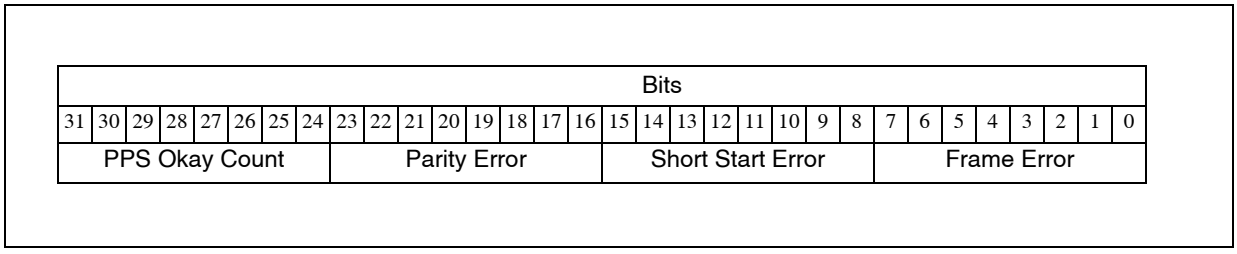


Figure A-49 RCIM IV GPS Receive Data Buffer

This is the GPS receive data buffer.

Offset: 04000 to 047FF

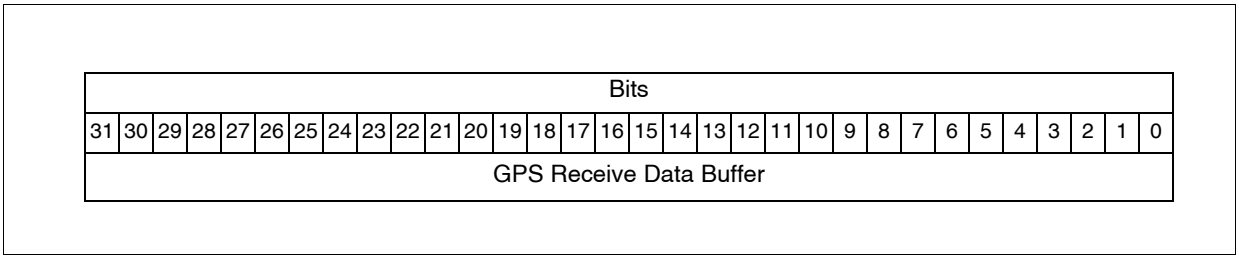


Figure A-50 RCIM IV GPS Transmit Data Buffer

This is the GPS transmit data buffer.

Offset: 04800 to 04FFF

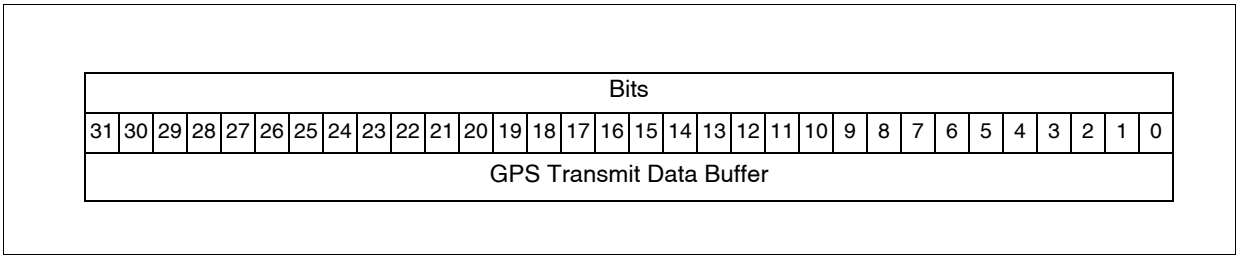


Figure A-51 RCIM IV IRIG Input Enable Register

Control whether the IRIG input signal is enabled or disabled (ignored).

Toggling the enable bit will cause the input interface to clear all programmed information and return to a reset state.

Offset: 06000

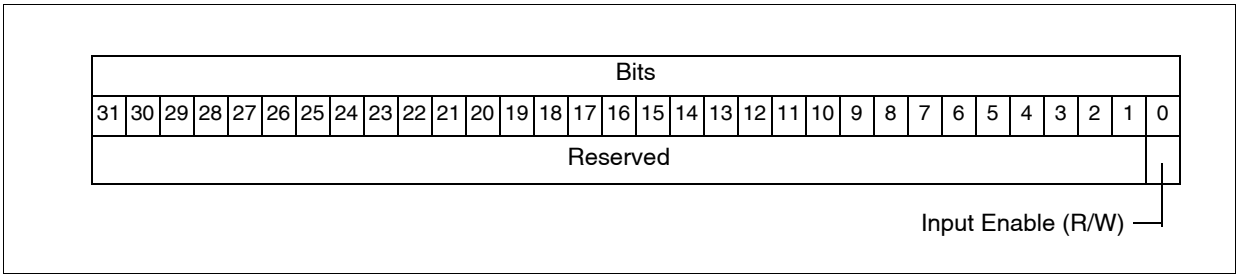
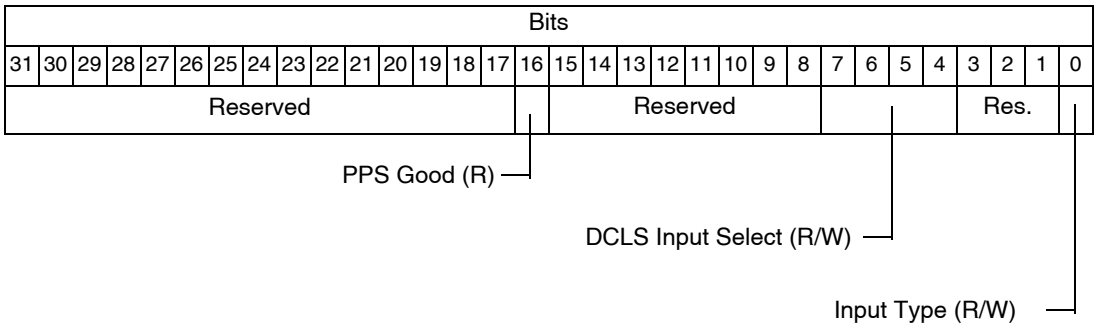


Figure A-52 RCIM IV IRIG Input Control Register

The IRIG input control register is used to configure IRIG input source attributes.

Offset: 06004



DCLS Input Select

Value	Input Source
0	External input 0
1	External input 1
2	External input 2
3	External input 3
4	External input 4
5	External input 5
6	External input 6
7	External input 7
8	External input 8
9	External input 9
10	External input 10
11	External input 11
12-14	Reserved
15	None (Default)

Input Type

Value	Input Source Type
0	Amplitude Modulation (AM)
1	DC Level Shift (DCLS)

NOTE

The AM input signal is on a dedicated line of the external I/O connector (EXT_AMIN). See Figure 3, “RCIM IV External Interrupt I/O Connector Pin-outs,” on page 2-6 for more information.

Figure A-53 RCIM IV IRIG Input Status Register

Determine the IRIG AM waveform high maximum value when the IRIG AM input type is selected.

Offsets: 06008

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																AM High Maximum (R)															

Multiply the AM High Maximum value by 375 microvolts to determine the peak value of the IRIG AM waveform.

Figure A-54 RCIM IV IRIG Input Error Register

The IRIG input error register contains information regarding communication errors with the IRIG module. All counts are reset to zero when IRIG input is enabled.

Offset: 0600C

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPS Okay Count (R)								AM Frame Error (R)								DCLS Frame Error (R)								Sync Error (R)							

PPS Okay Count	The number of times the IRIG PPS signal has indicated it is OK. This number should not change if the IRIG input data is reliable.
AM Frame Error	The number of AM framing errors that have occurred. Framing errors may occur if the source transmission stops and restarts, but should not occur when transmission is stable.
DCLS Frame Error	The number of DCLS framing errors that have occurred. Framing errors may occur if the source transmission stops and restarts, but should not occur once transmission is stable.
Sync Error	The number of sync errors that have occurred. Sync errors may occur if the source transmission stops and restarts, but should not occur once the transmission is stable.

Figure A-55 RCIM IV IRIG Input Seconds Register

This register contains the seconds field of the IRIG input data.

Offset: 06020

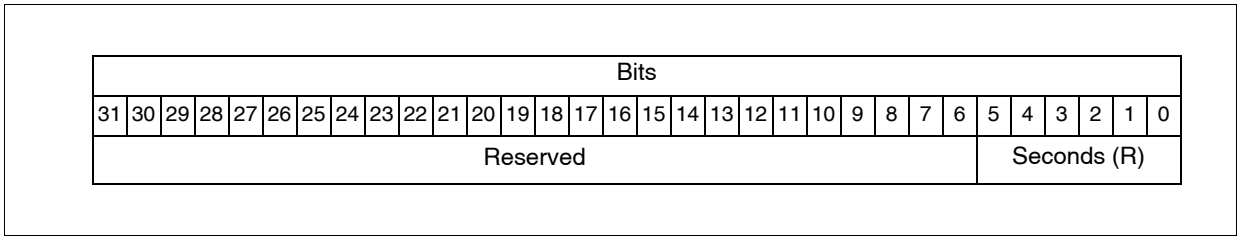


Figure A-56 RCIM IV IRIG Input Minutes Register

This register contains the minutes field of the IRIG input data.

Offset: 06024

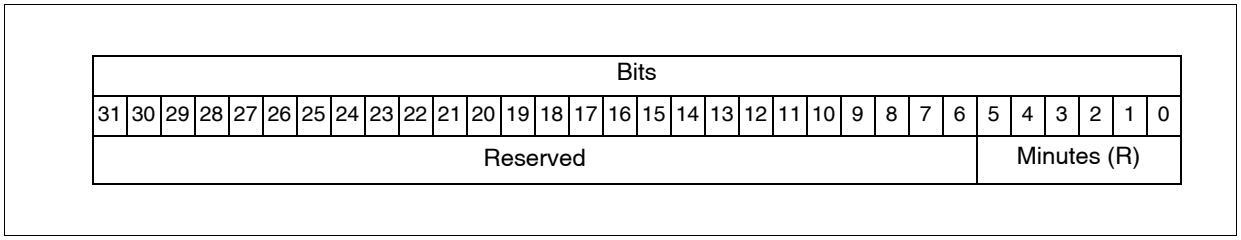


Figure A-57 RCIM IV IRIG Input Hours Register

This register contains the hours field of the IRIG input data.

Offset: 06028

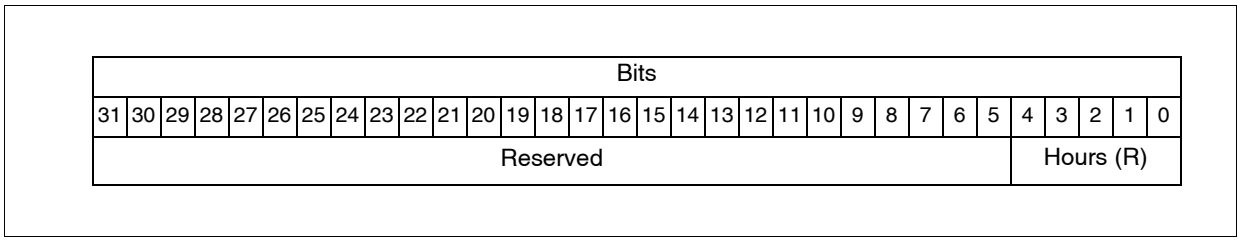


Figure A-58 RCIM IV IRIG Input Days Register

This register contains the days field of the IRIG input data.

Offset: 0602c

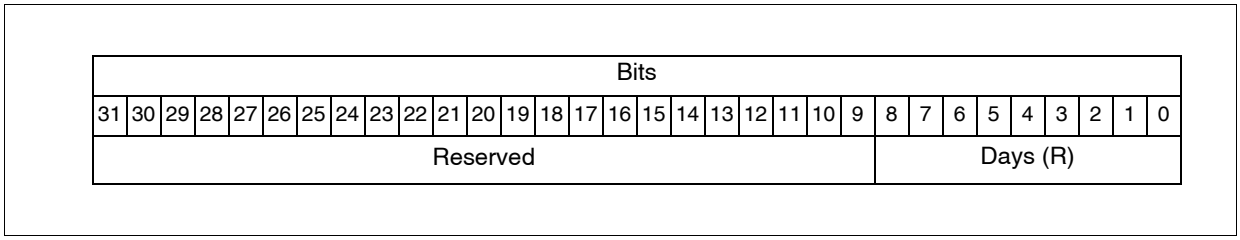


Figure A-59 RCIM IV IRIG Input Years Register

This register contains the years field of the IRIG input data.

Offset: 06030

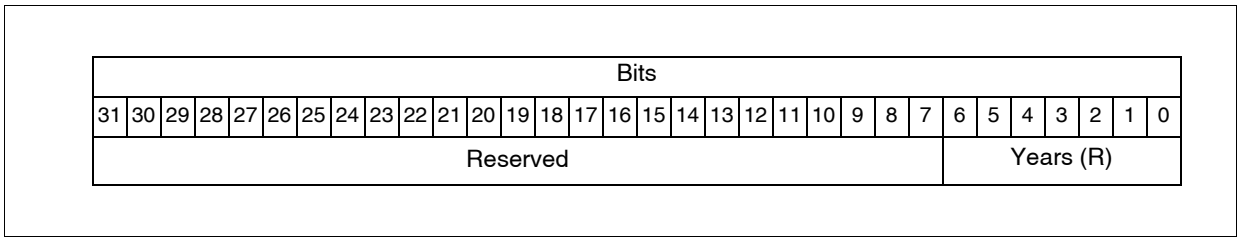


Figure A-60 RCIM IV IRIG Input Control Bits Register

This register contains the control bits field of the IRIG input data.

Offset: 06034

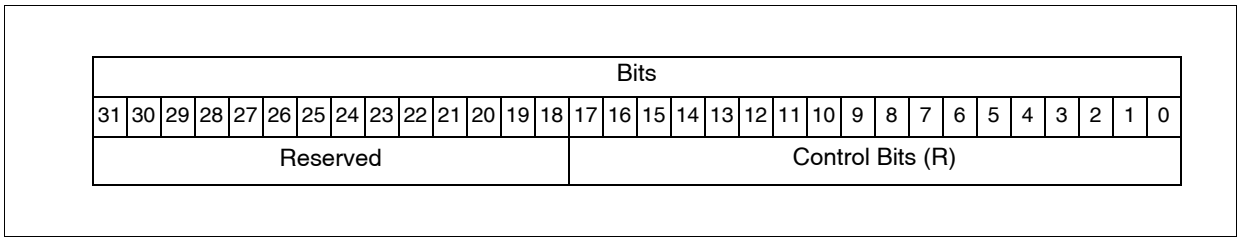


Figure A-61 RCIM IV IRIG Input SBS Register

This register contains the seconds of the day in straight binary seconds (SBS) field of the IRIG input data.

Offset: 06038

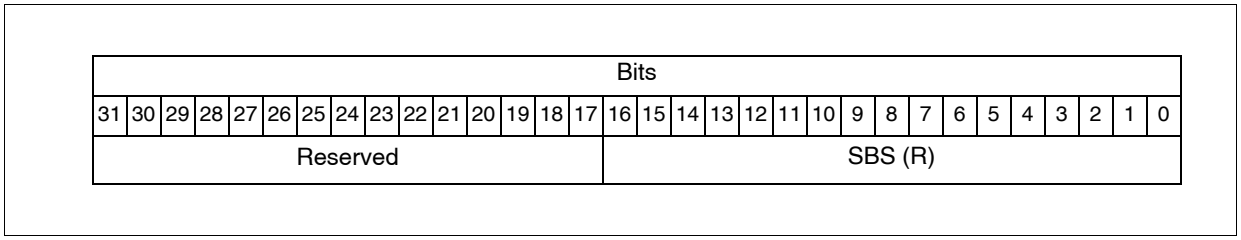


Figure A-62 RCIM IV IRIG Output Enable Register

Control whether the IRIG output signal is enabled or disabled.

Toggling the enable bit will cause the output interface to clear all programmed information and return to a reset state.

Offset: 06060

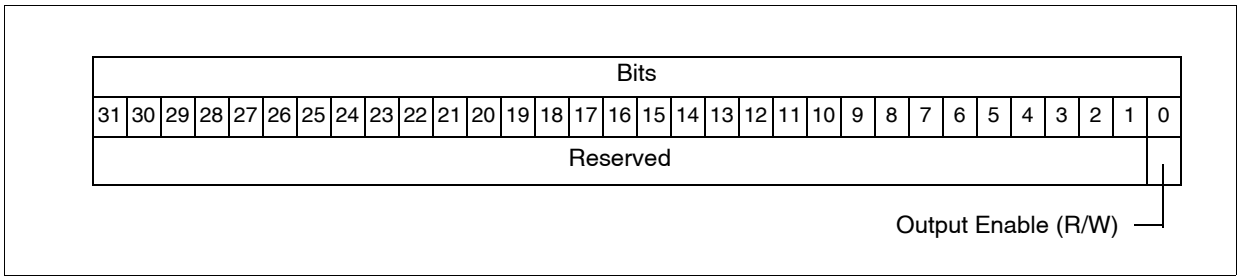
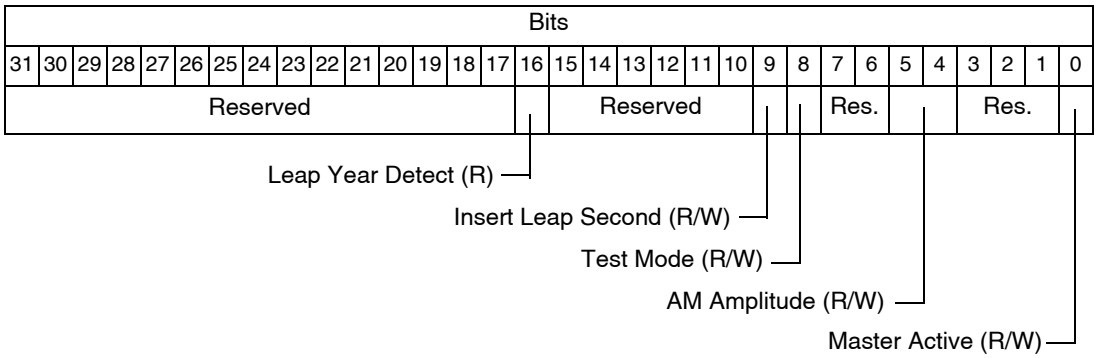


Figure A-63 RCIM IV IRIG Output Control Register

The IRIG output control register is used to configure IRIG output source attributes.

Offset: 06064



Leap Year Detect — Set by the hardware when the current year is a leap year.

Insert Leap Second — Set to manually add one leap second at the time indicated by the IRIG Output Leap Second Time Register.

NOTE

Manual insertion of leap seconds is unnecessary when the RCIM POSIX clock is being synchronized to system time by Chronyd utilizing RCIM GPS or an external timeserver.

Test Mode — Set to enable test mode in the factory. Not for general use.

AM Amplitude — Selects the AM amplitude peak output voltage as shown below:

AM Amplitude

Value	Output Voltage
0	2.5V peak-to-peak
1	5V peak-to-peak
2	8V peak-to-peak
3	Reserved

Master Active — Set to enable IRIG Master signal output transmission, and clear to disable transmission while the output register values are updated; the output register values will not update while transmission is disabled.

NOTE

It may take several cycles for any IRIG Slave devices to re-sync to the IRIG Master signal after the IRIG Master signal output transmission is activated.

Figure A-64 RCIM IV IRIG Output Adust Register

This register holds the IRIG output offset adjustment used to compensate for various timing delays.

Offset: 06068

Bits																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																				Output Adjust (R/W)											

Output Adjust	This value can be used to offset the IRIG output timing to compensate for delays due to AM modulation/demodulation and cabling. The offset is in the negative direction causing the IRIG output data to be sent earlier. The formula for the compensation value is: Output Adjust = (*desired offset in nanoseconds* / 400) * 25. The maximum offset is 65.52 microseconds in 400 nanosecond increments.

Figure A-65 RCIM IV IRIG Output Leap Second Time Register

This register holds the IRIG output offset adjustment used to compensate for various timing delays.

Offset: 0606C

Bits																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R.	Leap Second Year (R/W)							Reserved							Leap Second Day (R/W)									Res.			Leap Second Hour (R/W)				

Leap Second Year	The year that the leap second will be inserted.
Leap Second Day	The day that the leap second will be inserted.
Leap Second Hour	The hour that the leap second will be inserted.

Figure A-66 RCIM IV IRIG Output Seconds Register

This register contains the seconds field of the IRIG output data.

Offset: 06080

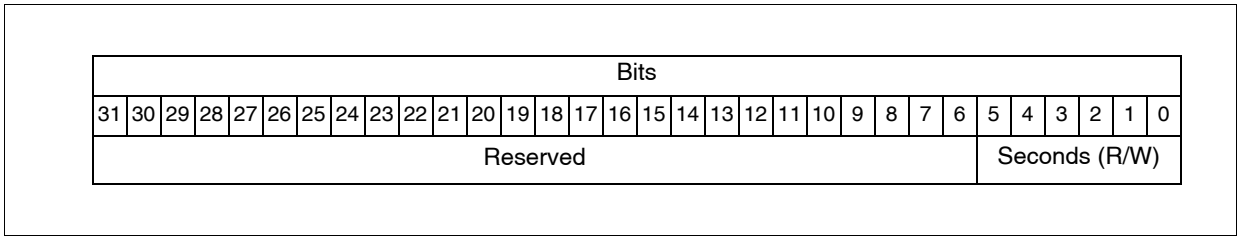


Figure A-67 RCIM IV IRIG Output Minutes Register

This register contains the minutes field of the IRIG output data.

Offset: 06084

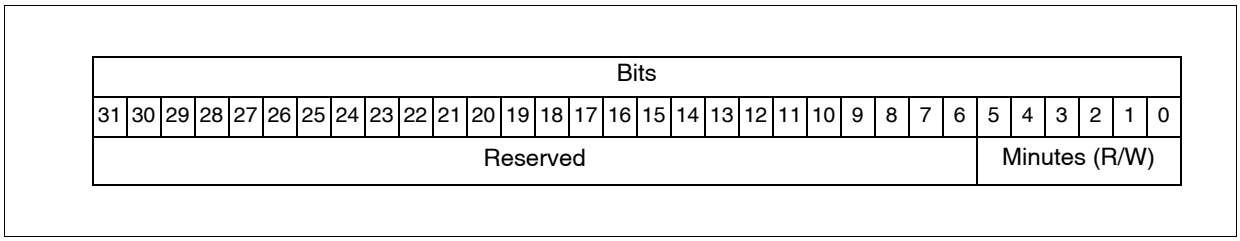


Figure A-68 RCIM IV IRIG Output Hours Register

This register contains the hours field of the IRIG output data.

Offset: 06088

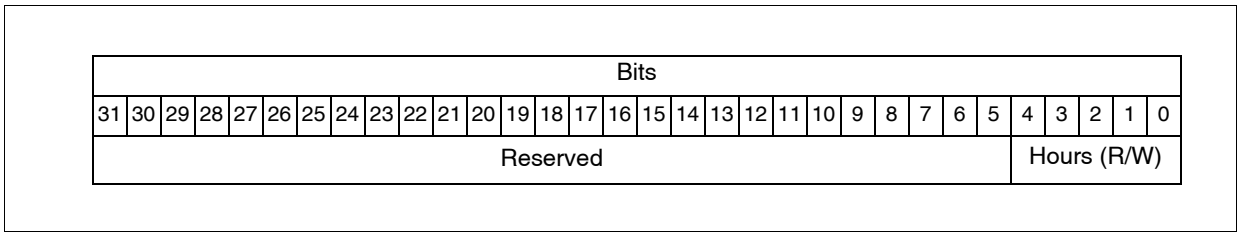


Figure A-69 RCIM IV IRIG Output Days Register

This register contains the days field of the IRIG output data.

Offset: 0608c

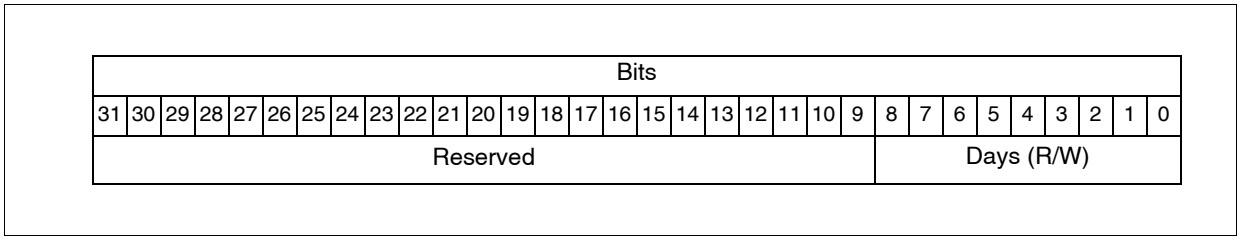


Figure A-70 RCIM IV IRIG Output Years Register

This register contains the years field of the IRIG output data.

Offset: 06090

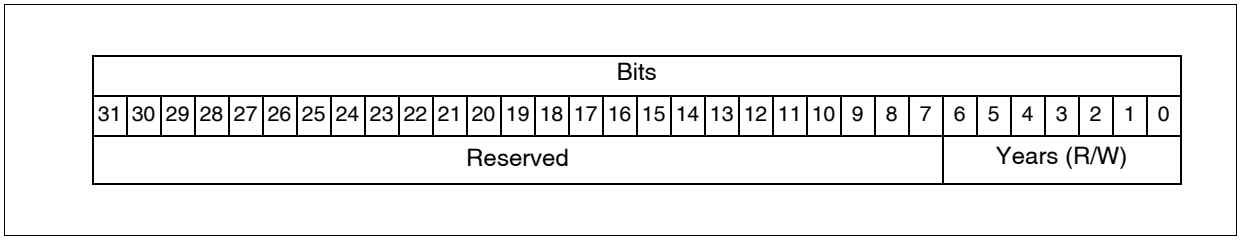


Figure A-71 RCIM IV IRIG Output Control Bits Register

This register contains the control bits field of the IRIG output data.

Offset: 06094

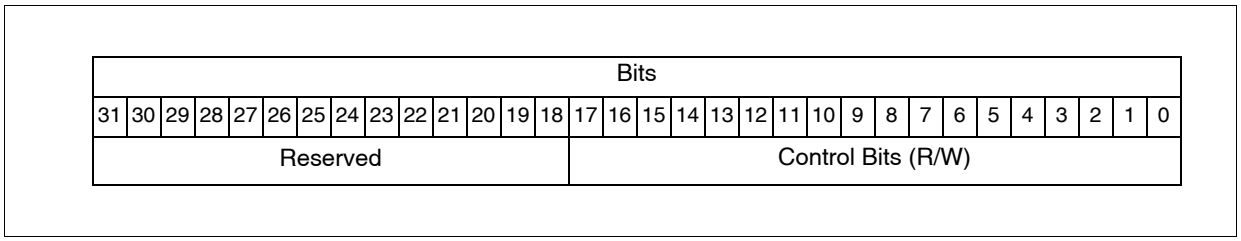
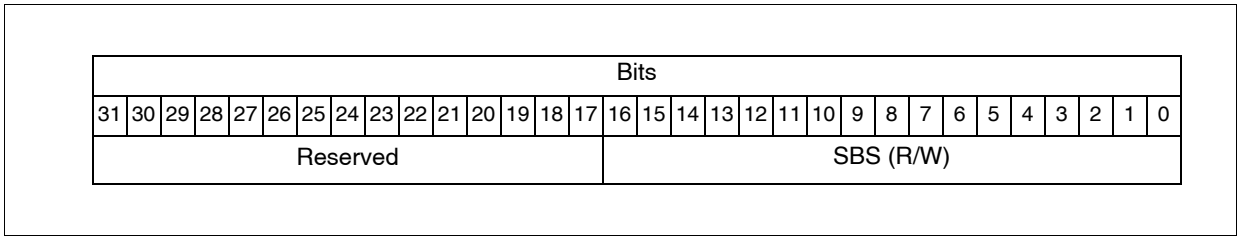


Figure A-72 RCIM IV IRIG Output SBS Register

This register contains the seconds of the day in straight binary seconds (SBS) field of the IRIG output data.

Offset: 06098



B
RCIM III Registers

This section contains the address map and registers on the RCIM III board.

Note that some registers appear at two places in the physical address space. For these registers there is an associated *Xregister*. For example PCSAT and XPCSAT. The *Xregister* accommodates systems with a 64k, rather than the old 4k page size.

RCIM III Address Map

Address	Function	
0xXXX00000	Board Status/Control Register	
0xXXX00004	FirmwareRev/OptionsPresentRegister	
0xXXX00010	Interrupt Enable Register #1	
0xXXX00014	Interrupt Enable Register #2	
0xXXX00020	Interrupt Request Register #1	(Write Only)
0xXXX00024	Interrupt Request Register #2	(Write Only)
0xXXX00020	Interrupt Pending Register #1	(Read Only)
0xXXX00024	Interrupt Pending Register #2	(Read Only)
0xXXX00030	Interrupt Clear Register #1	
0xXXX00034	Interrupt Clear Register #2	
0xXXX00040	Interrupt Arm Register #1	
0xXXX00044	Interrupt Arm Register #2	
0xXXX00050	Interrupt Select Level Register #1	
0xXXX00054	Interrupt Select Level Register #2	
0xXXX00060	Interrupt Select Polarity Register #1	
0xXXX00064	Interrupt Select Polarity Register #2	
0xXXX00070	External Interrupt Routing Register #1	
0xXXX00074	External Interrupt Routing Register #2	
0xXXX00078	External Interrupt Routing Register #3	
0xXXX00080	Cable Interrupt Routing Register #1	
0xXXX00084	Cable Interrupt Routing Register #2	
0xXXX00088	Cable Interrupt Routing Register #3	
0xXXX00200	PPS Snapshot Register	(Read Only)
0xXXX00210	Cable Snapshot Register	(Read Only)
0xXXX00220	Cable Master Time Register	(Read Only)
0xXXX00400	Clear Cable Errors	(Write Only)
0xXXX00410	Output Cable Status Register	(Read Only)
0xXXX00420	Input Cable Status Register	(Read Only)
0xXXX01000 0xXXX10000	Tick Clock Upper Tick Clock Upper	
0xXXX01008 0xXXX10008	Tick Clock Lower Tick Clock Lower	

Address	Function	
0xXXX01010 0xXXX10010	Tick Clock Status/Control Tick Clock Status/Control	
0xXXX01100 0xXXX10100	POSIX Clock Seconds POSIX Clock Seconds	
0xXXX01108 0xXXX10108	POSIX Clock Nanoseconds POSIX Clock Nanoseconds	
0xXXX01110 0xXXX10110	POSIX Clock Status/Control POSIX Clock Status/Control	
0xXXX01114 0xXXX10114	POSIX Clock Skip/Add Time POSIX Clock Skip/Add Time	(Write Only) (Write Only)
0xXXX01120 0xXXX10120	Clock Frequency Adjust Register Clock Frequency Adjust Register	
0xXXX02000	RTC #0 Control	
0xXXX02010	RTC #0 Timer	
0xXXX02014	RTC #0 Repeat	
0xXXX02020	RTC #1 Control	
0xXXX02030	RTC #1 Timer	
0xXXX02034	RTC #1 Repeat	
0xXXX02040	RTC #2 Control	
0xXXX02050	RTC #2 Timer	
0xXXX02054	RTC #2 Repeat	
0xXXX02060	RTC #3 Control	
0xXXX02070	RTC #3 Timer	
0xXXX02074	RTC #3 Repeat	
0xXXX02080	RTC #4 Control	
0xXXX02090	RTC #4 Timer	
0xXXX02094	RTC #4 Repeat	
0xXXX020A0	RTC #5 Control	
0xXXX020B0	RTC #5 Timer	
0xXXX020B4	RTC #5 Repeat	
0xXXX020C0	RTC #6 Control	
0xXXX020D0	RTC #6 Timer	
0xXXX020D4	RTC #6 Repeat	
0xXXX020E0	RTC #7 Control	
0xXXX020F0	RTC #7 Timer	
0xXXX020F4	RTC #7 Repeat	
0xXXX03000 0xXXX30000	Programmable Interrupt Generator Programmable Interrupt Generator	
0xXXX03010 0xXXX30010	Programmable INTR Generator Set Programmable INTR Generator Set	(Write Only) (Write Only)
0xXXX03020 0xXXX30020	Programmable INTR Generator Clear Programmable INTR Generator Clear	(Write Only) (Write Only)
0xXXX03100	SPI Count Register	
0xXXX03200	GPS Receive Pointers	
0xXXX03204	GPS Transmit Pointers	
0xXXX03208	GPS Debug Control Register	

Address	Function	
0xXXX0320C	GPS Communication Error Register	
0xXXX03800- 0xXXX03FFF	SPI Data Buffer	
0xXXX04000- 0xXXX047FF	GPS Receive Data Buffer	
0xXXX04800- 0xXXX04FFF	GPS Transmit Data Buffer	

RCIM III Registers

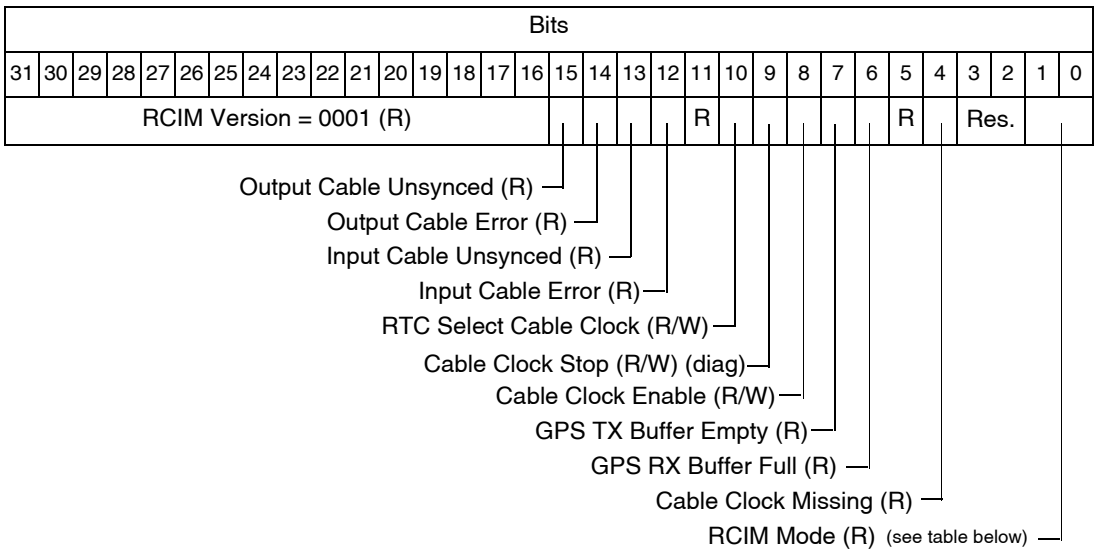
RCIM III registers are illustrated in this section.

NOTE: Unless otherwise stated, a bit value of 1=on; 0=off

Figure B-1 RCIM III Board Status/Control Register

This register provides status and control of certain features of the RCIM III board.

Offset: 00000



RCIM Mode Bit Values

Bits 1 0	Description	Input Cable	Output Cable	Cable Clock
1 1	Stand-alone Master RCIM	Open	Open	Inactive
1 0	Master RCIM	Open	Connected	Sourcing
0 1	Last Slave RCIM	Connected	Connected	Receiving
0 0	Pass-thru Slave RCIM	Connected	Open	Receiving

Figure B-2 RCIM III Firmware Revision/Options Present Register

This register provides information on what options are present on this RCIM board and the firmware revision.

Offset: 000004

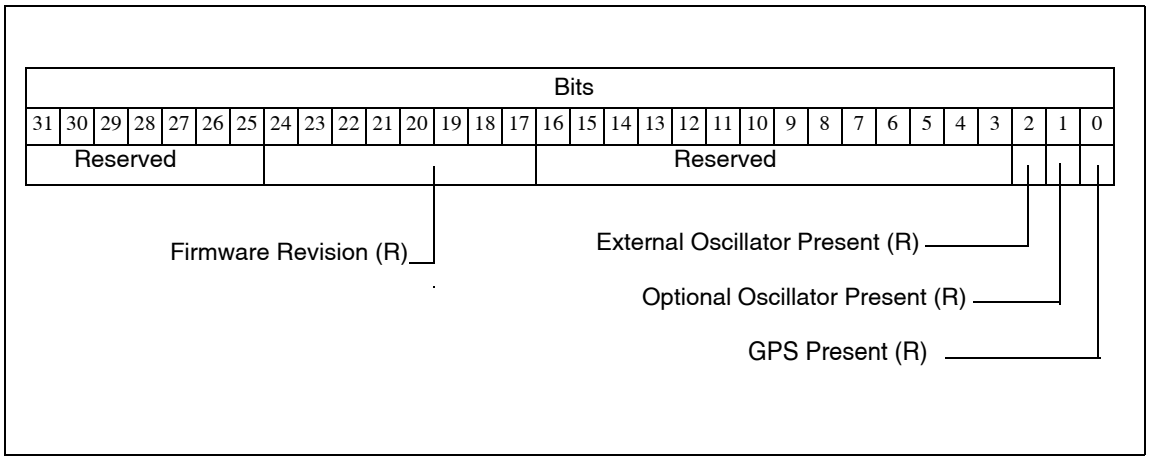


Figure B-3 RCIM III Interrupt Enable/Request/Pending/Clear/Arm/Level/Polarity Registers

The enable registers enable the selected interrupts.
The request registers are software driven requests of the selected interrupts.
The pending registers are pending requests.
The clear registers clear the selected interrupts.
The arm registers arm the selected interrupts for edge triggering.
The level registers set level (1) or edge (0) for the selected interrupts.
The polarity registers set polarity high (1) or low (0) for the selected interrupts.

Offsets: 00010, 00014, 00020, 00024, 00030, 00034, 00040, 00044, 00050, 00054, 00060, 00064

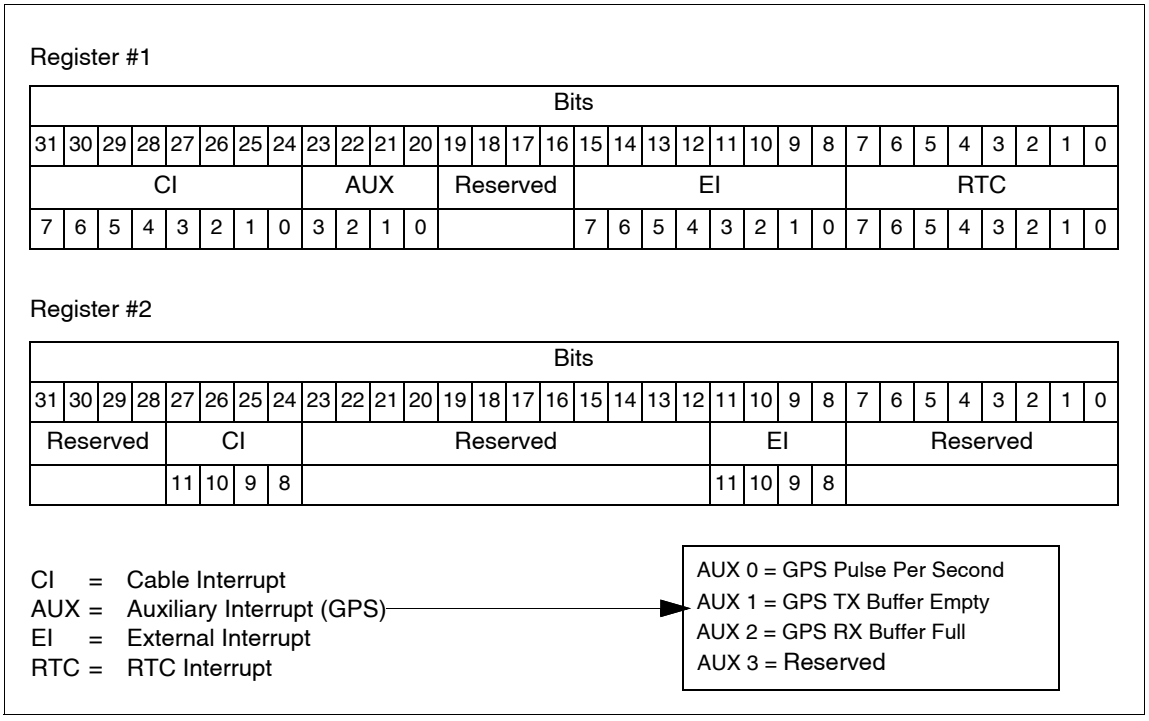
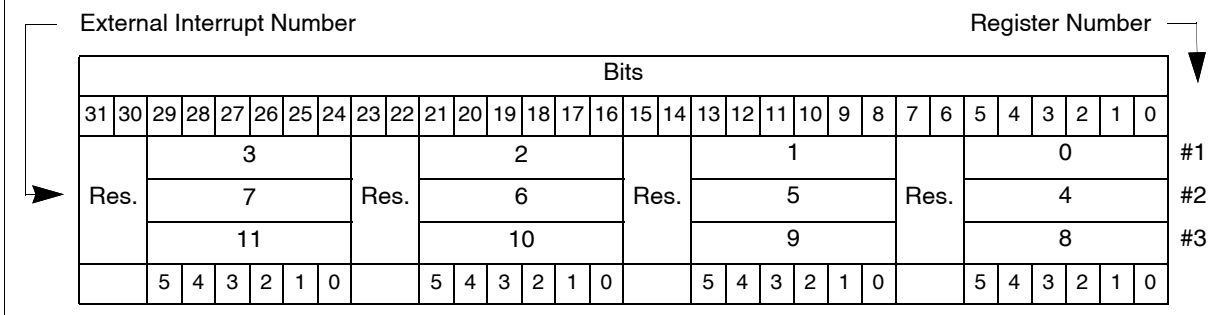


Figure B-4 RCIM III External Interrupt Routing Registers

The external interrupt routing registers route selected interrupts to the external interrupt connector.

Offset: 00070, 00074, 00078



Value	Description
0x00	Reserved
0x01	Reserved
0x02	Reserved
0x03	Reserved
0x04	RTC 0
0x05	RTC 1
0x06	RTC 2
0x07	RTC 3
0x08	Edge Interrupt 0
0x09	Edge Interrupt 1
0x0A	Edge Interrupt 2
0x0B	Edge Interrupt 3
0x0C	PIG 0
0x0D	PIG 1
0x0E	PIG 2
0x0F	PIG 3
0x10	Cable Interrupt 0
0x11	Cable Interrupt 1
0x12	Cable Interrupt 2
0x13	Cable Interrupt 3
0x14	Cable Interrupt 4
0x15	Cable Interrupt 5
0x16	Cable Interrupt 6
0x17	Cable Interrupt 7
0x18	Cable Interrupt 8
0x19	Cable Interrupt 9
0x1A	Cable Interrupt 10
0x1B	Cable Interrupt 11
0x1C	Reserved
0x1D	Reserved
0x1E	Reserved
0x1F	Reserved

Value	Description
0x20	GPS_PPS
0x21	Reserved
0x22	Reserved
0x23	Reserved
0x24	RTC 4
0x25	RTC 5
0x26	RTC 6
0x27	RTC 7
0x28	Edge Interrupt 4
0x29	Edge Interrupt 5
0x2A	Edge Interrupt 6
0x2B	Edge Interrupt 7
0x2C	PIG 4
0x2D	PIG 5
0x2E	PIG 6
0x2F	PIG 7
0x30	Reserved
0x31	Reserved
0x32	Reserved
0x33	Reserved
0x34	Reserved
0x35	Reserved
0x36	Reserved
0x37	Reserved
0x38	Edge Interrupt 8
0x39	Edge Interrupt 9
0x3A	Edge Interrupt 10
0x3B	Edge Interrupt 11
0x3C	PIG 8
0x3D	PIG 9
0x3E	PIG 10
0x3F	PIG 11

Figure B-5 RCIM III Cable Interrupt Routing Registers

The cable interrupt routing registers route selected interrupts to the RCIM interconnecting cable.

Offsets: 00080, 00084, 00088

Cable Interrupt Register 1

Cable Interrupt Number

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
7				6				5				4				3				2				1				0			
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

Cable Interrupt Register 2

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved																11				10				9				8			
																3	2	1	0	3	2	1	0	3	2	1	0	3	2	1	0

Cable Interrupt Register 3

Bits																													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								11		10		9		8		7		6		5		4		3		2		1		0	
								5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4	5	4

Value	Description	Value	Description	Value	Description
0x00	Reserved	0x10	Reserved	0x20	GPS_PPS
0x01	Reserved	0x11	Reserved	0x21	Reserved
0x02	Reserved	0x12	Reserved	|	|
0x03	Reserved	0x13	Reserved	0x27	Reserved
0x04	RTC 0	0x14	RTC 4	0x28	Edge Interrupt 8
0x05	RTC 1	0x15	RTC 5	0x29	Edge Interrupt 9
0x06	RTC 2	0x16	RTC 6	0x2A	Edge Interrupt 10
0x07	RTC 3	0x17	RTC 7	0x2B	Edge Interrupt 11
0x08	Edge Interrupt 0	0x18	Edge Interrupt 4	0x2C	PIG 8
0x09	Edge Interrupt 1	0x19	Edge Interrupt 5	0x2D	PIG 9
0x0A	Edge Interrupt 2	0x1A	Edge Interrupt 6	0x2E	PIG 10
0x0B	Edge Interrupt 3	0x1B	Edge Interrupt 7	0x2F	PIG 11
0x0C	PIG 0	0x1C	PIG 4	0x30	Reserved
0x0D	PIG 1	0x1D	PIG 5	|	|
0x0E	PIG 2	0x1E	PIG 6	|	|
0x0F	PIG 3	0x1F	PIG 7	0x3F	Reserved

Figure B-6 RCIM III PPS Snapshot Register

The PPS Snapshot register contains a snapshot of the nanoseconds field and two bits of the seconds field of the POSIX clock. The snapshot is taken every time the GPS PPS signal occurs.

Offset: 00200

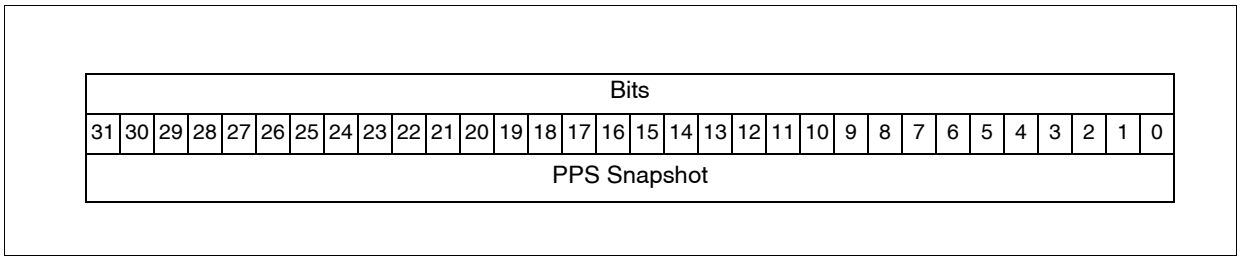


Figure B-7 RCIM III Cable Snapshot Register

The Cable Snapshot register contains a snapshot of the nanoseconds field and two bits of the seconds field of the POSIX clock. The snapshot is taken every time the cable master time is received.

Offset: 00210

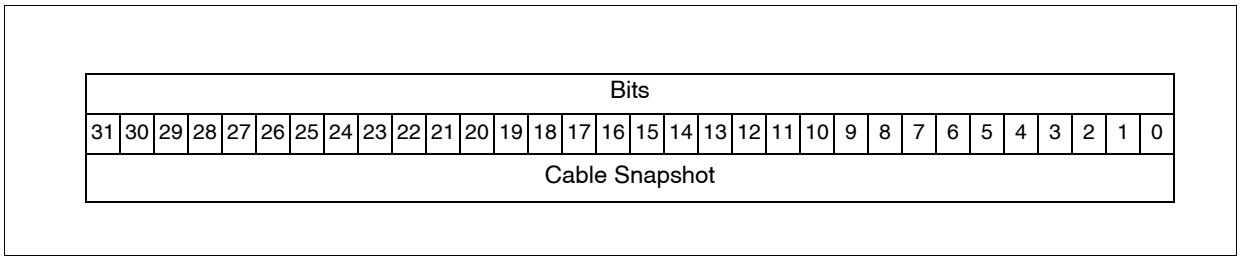


Figure B-8 RCIM III Cable Master Time Register

The Cable Master Time register contains the seconds field of the master RCIM POSIX clock that is transmitted on the cable at every transition of the clock at the seconds boundary.

Offset: 00220

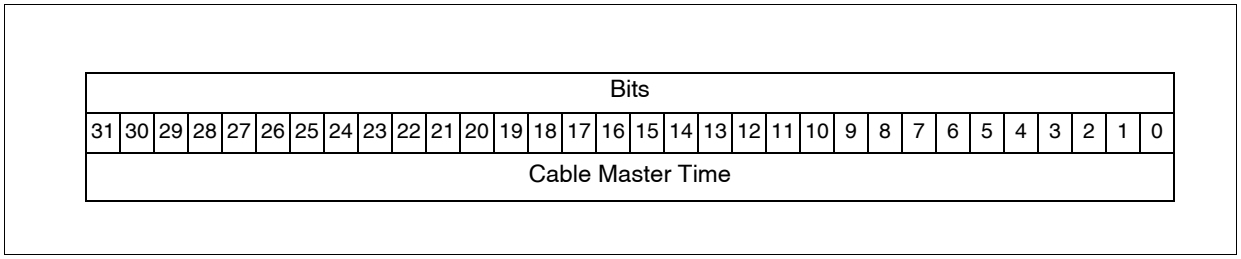


Figure B-9 RCIM III Clear Cable Errors Register

This is a Write Only register that clears any reported cable errors. The data field is don't care.

Offset: 00400

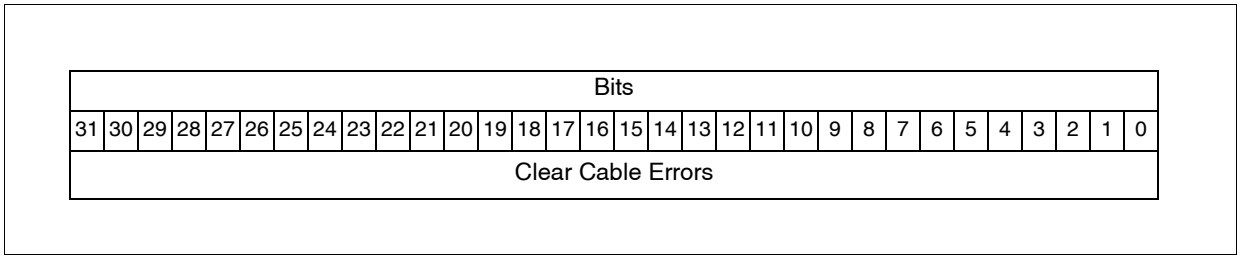


Figure B-10 RCIM III Output Cable Status Register

This register provides detailed hardware status information pertaining to the output cable.

Offset: 00410

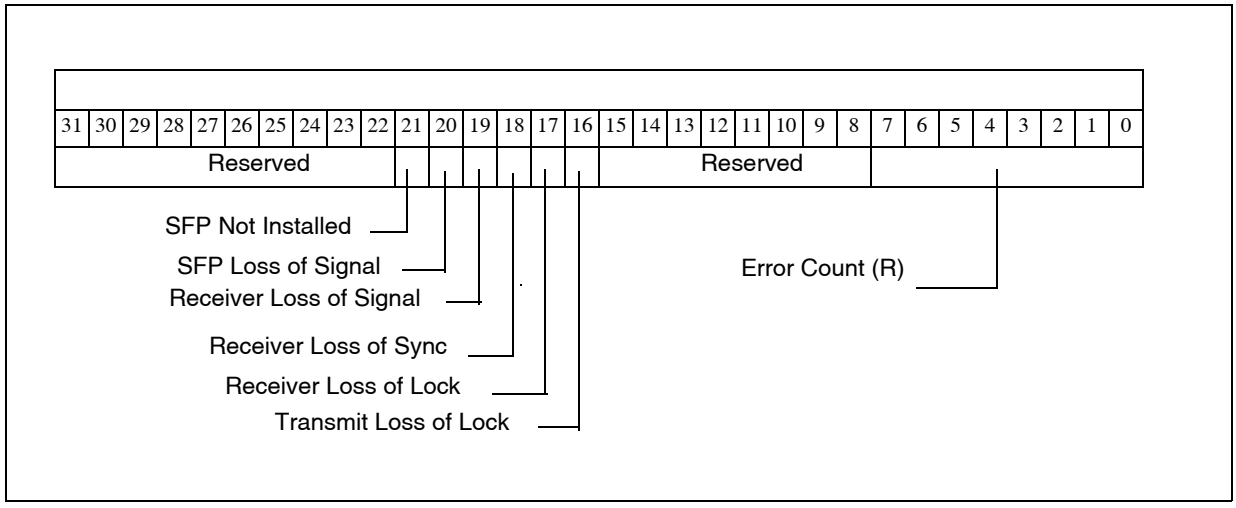


Figure B-11 RCIM III Input Cable Status Register

This register provides detailed hardware status information pertaining to the input cable.

Offset: 00420

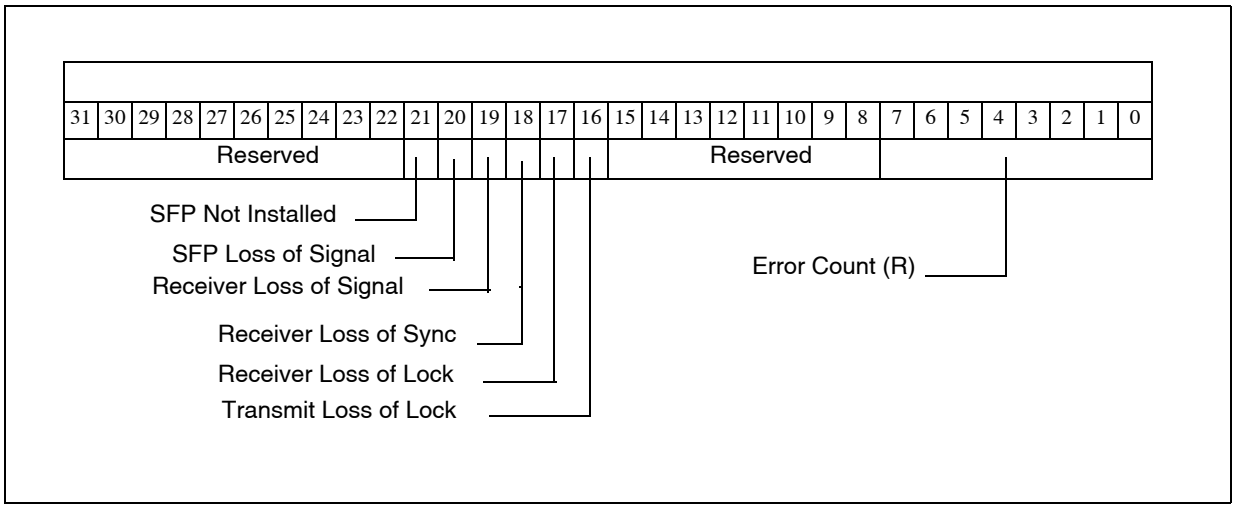


Figure B-12 RCIM III Tick Clock Upper Register

This register contains the upper 32 bits of the tick clock.

Offsets: 01000, 10000

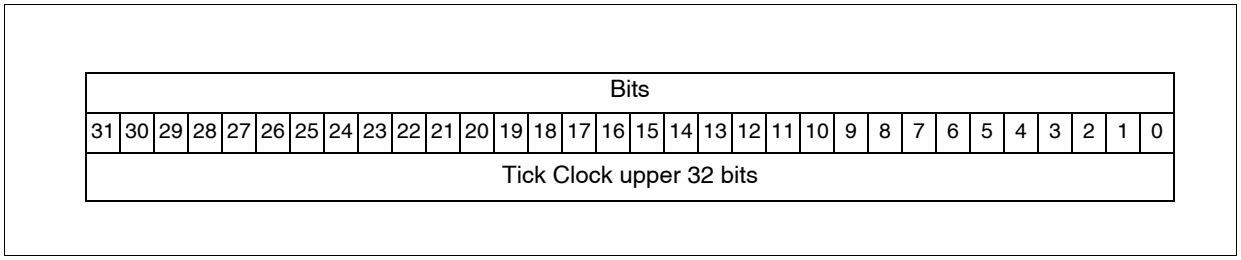


Figure B-13 RCIM III Tick Clock Lower Register

This register contains the lower 32 bits of the tick clock.

Offsets: 01008, 10008

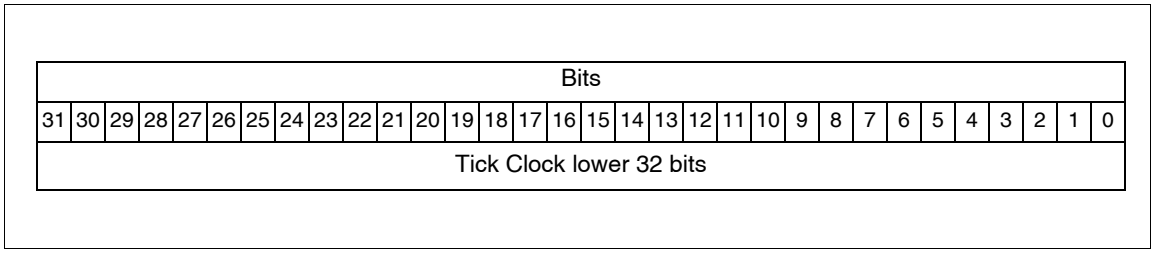


Figure B-14 RCIM III Tick Clock Status/Control Register

This register provides status and control of the tick clock.

Offsets: 01010, 10010

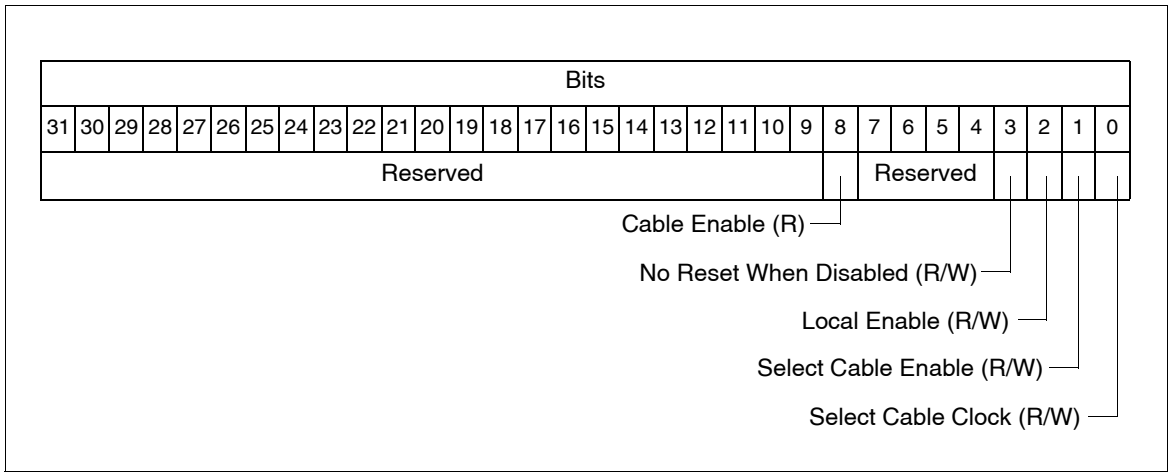


Figure B-15 RCIM III POSIX Clock Seconds Register

This register contains the POSIX clock seconds.

Offsets: 01100, 10100

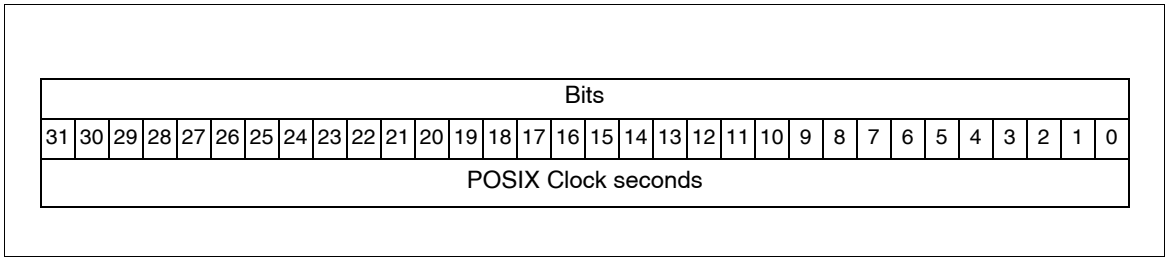


Figure B-16 RCIM III POSIX Clock Nanoseconds Register

This register contains the POSIX clock nanoseconds.

Offsets: 01108, 10108

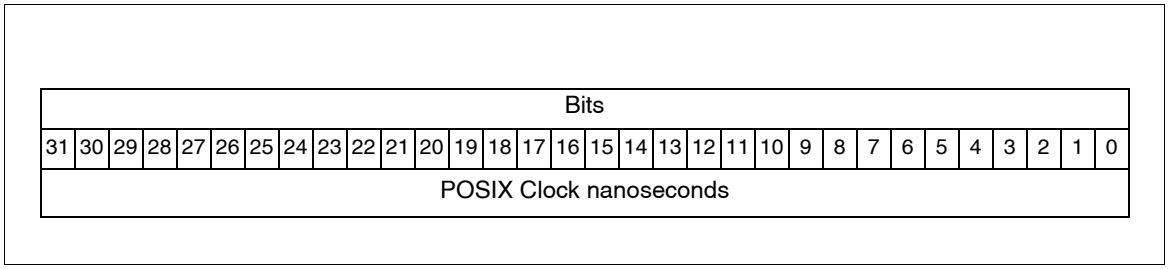


Figure B-17 RCIM III POSIX Clock Status/Control Register

This register provides status and control of the POSIX clock.

Offsets: 01110, 10110

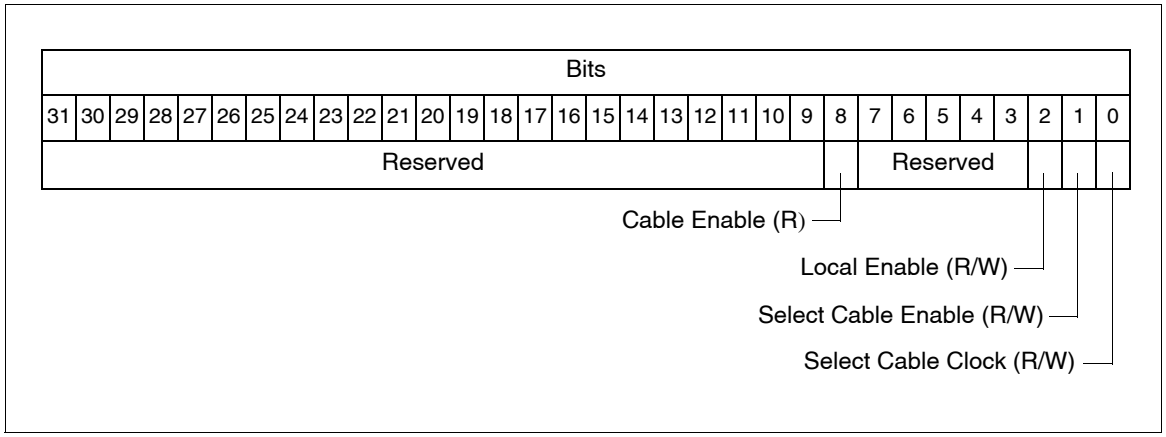


Figure B-18 RCIM III POSIX Clock Skip/Add Time Register

This register skips/adds time to the POSIX clock in 400 nanosecond increments.

Offsets: 01114, 10114

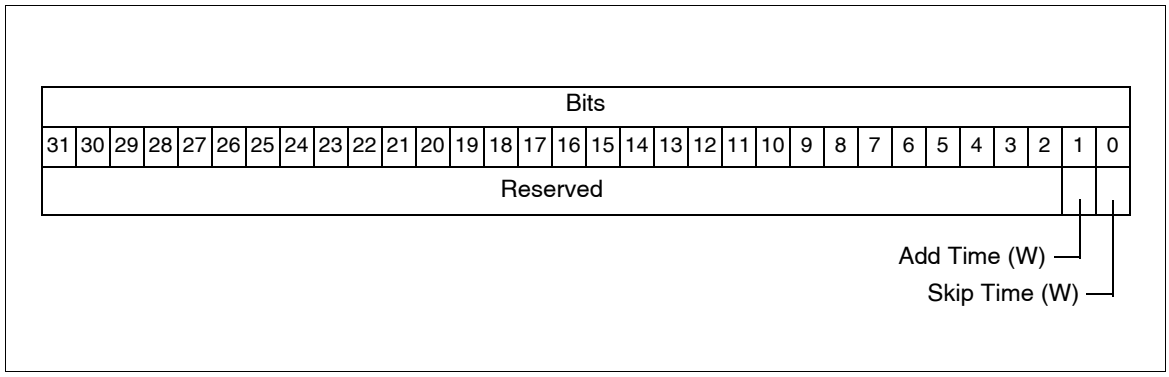


Figure B-19 RCIM III Clock Frequency Adjust Register

The clock frequency adjust register is used to control the frequency of the 10 MHz master clock.

Offsets: 01120, 10120

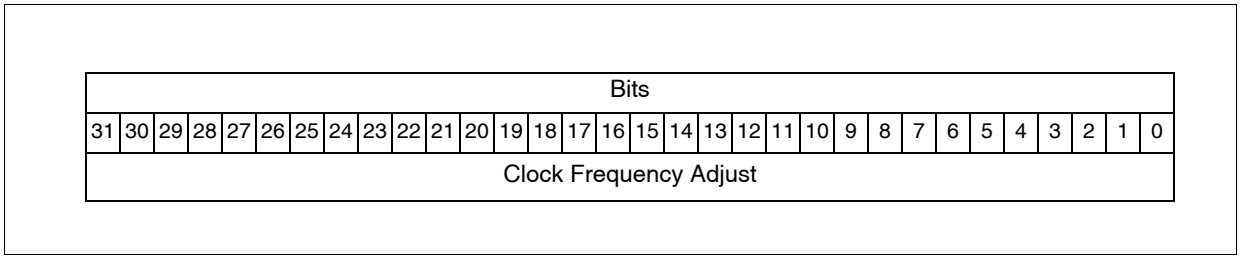


Figure B-20 RCIM III RTC Timer Registers

The initial RTC timer value is loaded in the RTC timer registers. The current value of the timer is read from this register. NOTE: Loading this register also loads the RTC Repeat Register for compatibility with RCIM.

Offsets: 02010, 02030, 02050, 02070, 02090, 020B0, 020D0, 020F0

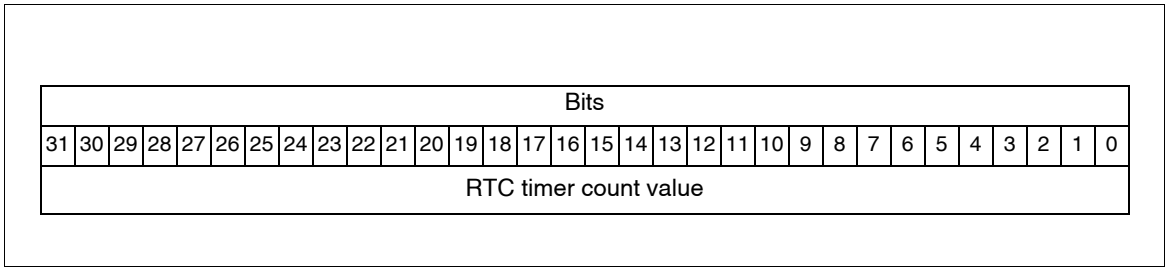


Figure B-21 RCIM III RTC Repeat Registers

The RTC repeat registers contain the repeat count value.

Offsets: 02014, 02034, 02054, 02074, 02094, 020B4, 020D4, 020F4

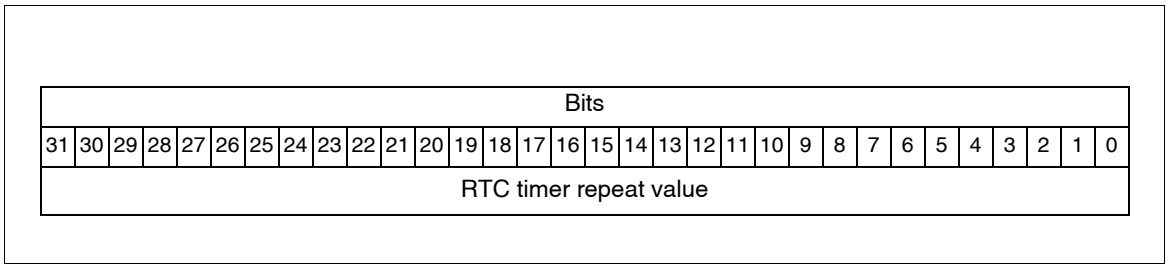
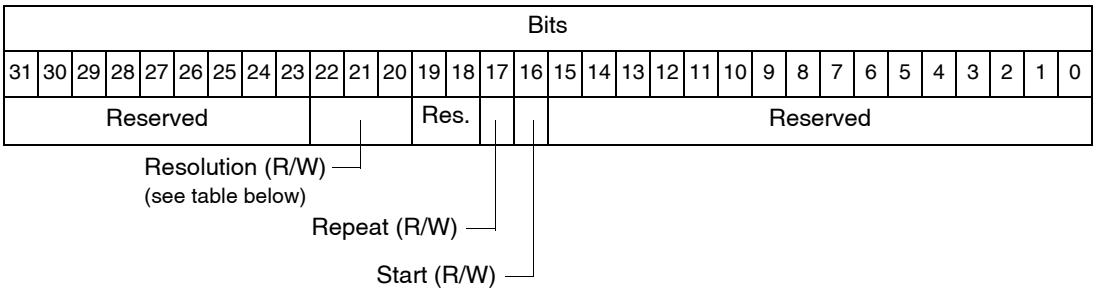


Figure B-22 RCIM III RTC Control Registers

This register provides control of the RTCs.

Offsets: 02000, 02020, 02040, 02060, 02080, 020A0, 020C0, 020E0



Resolution Bit Values

Bits 22	21	20	Description
0	0	0	1 microsecond
0	0	1	10 microseconds
0	1	0	100 microseconds
0	1	1	1 millisecond
1	0	0	10 milliseconds
1	0	1	stopped
1	1	0	stopped
1	1	1	stopped

Figure B-23 RCIM III Programmable Interrupt Generator Register

This register identifies the programmable interrupts.

Offsets: 03000, 30000

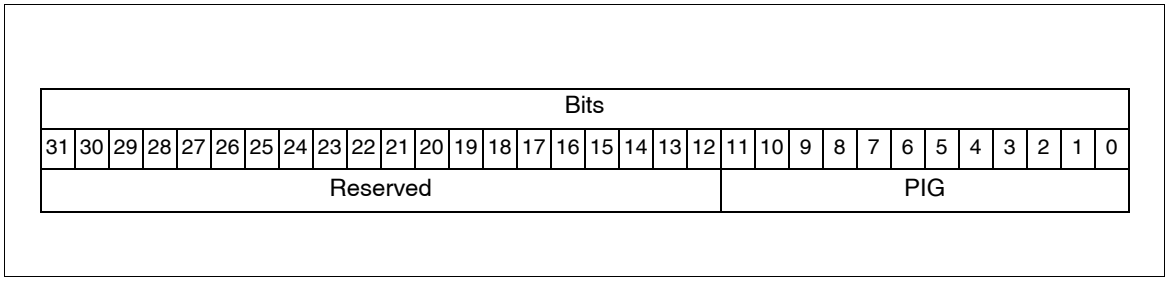


Figure B-24 RCIM III Programmable Interrupt Set and Clear Registers

Writing to these registers sets/clears the unitary bits in the Programmable Interrupt Register without affecting the other bits.

Offsets: 03010, 30010 03020, 30020

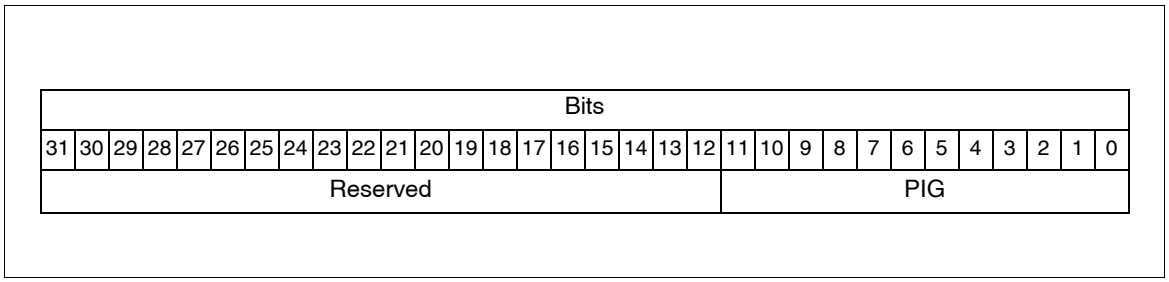


Figure B-25 RCIM III GPS Receive Pointers

The GPS receive pointers are used for communication with the optional GPS module.

Offset: 03200

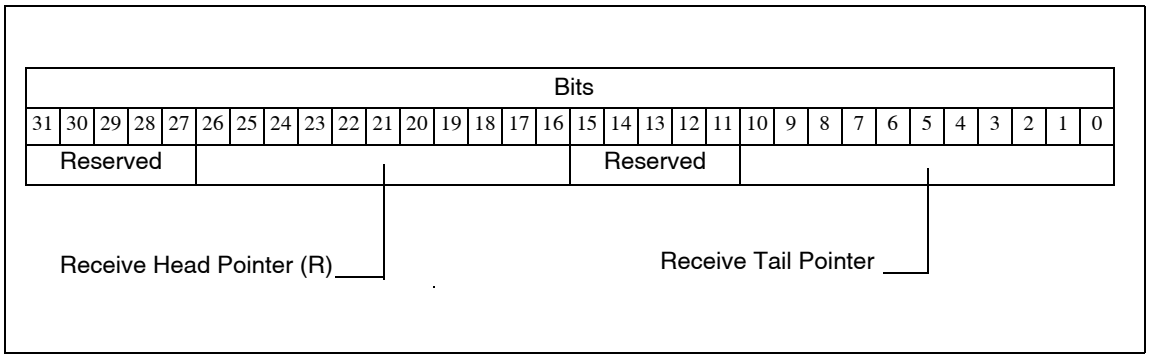


Figure B-26 RCIM III GPS Transmit Pointers

The GPS transmit pointers are used for communication with the optional GPS module.

Offset: 03204

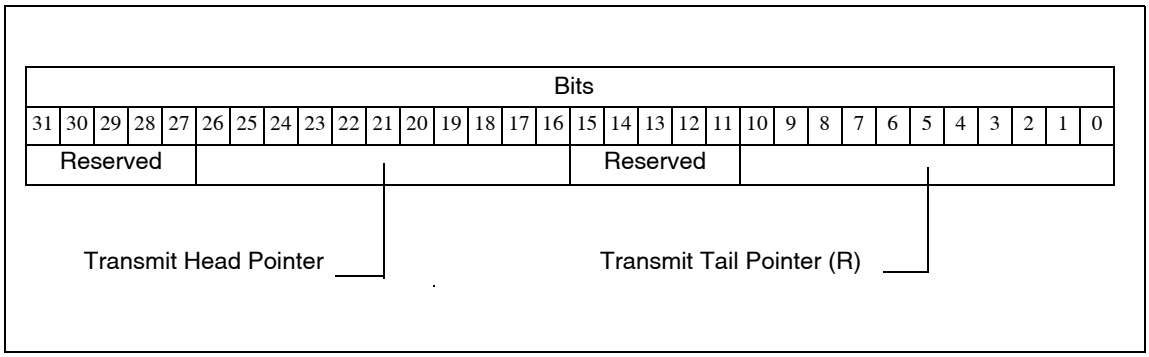


Figure B-27 RCIM III GPS Debug Control Register

The GPS debug control register contains bits used during testing and debug. Setting any of these bits will disable RCIM communication with the GPS module.

Offset: 03208

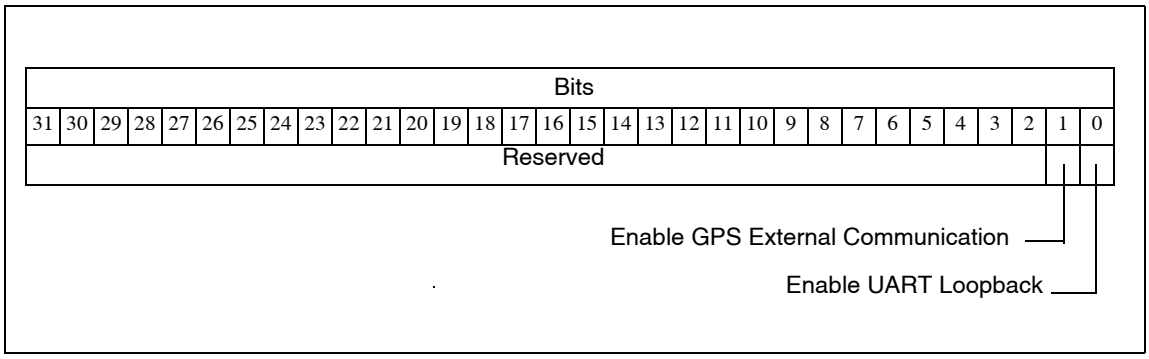


Figure B-28 RCIM III GPS Communication Error Register

The GPS communication error register contains counts of communication errors with the GPS module.

Any write to this register will reset the communication interface to the GPS module.

Offset: 0320C

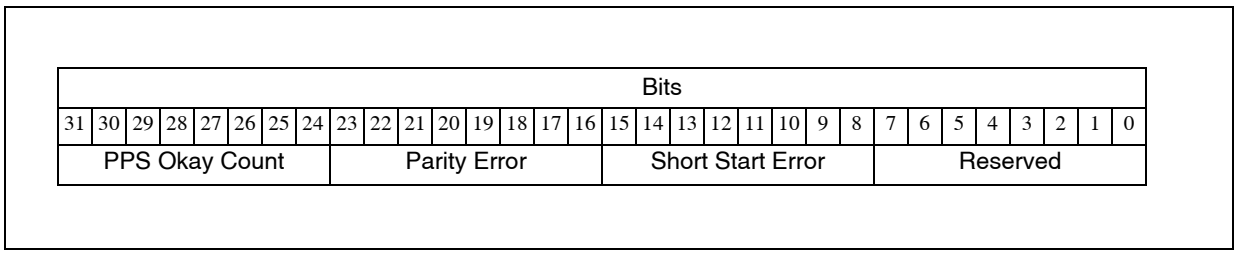


Figure B-29 RCIM III GPS Receive Data Buffer

This is the GPS receive data buffer.

Offset: 04000 to 047FF

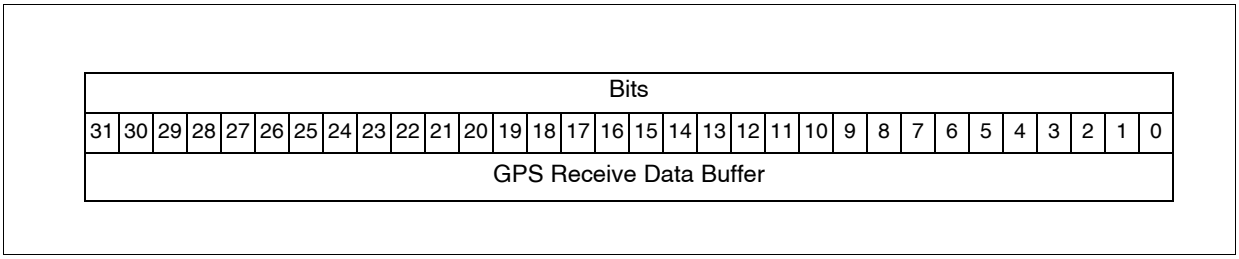
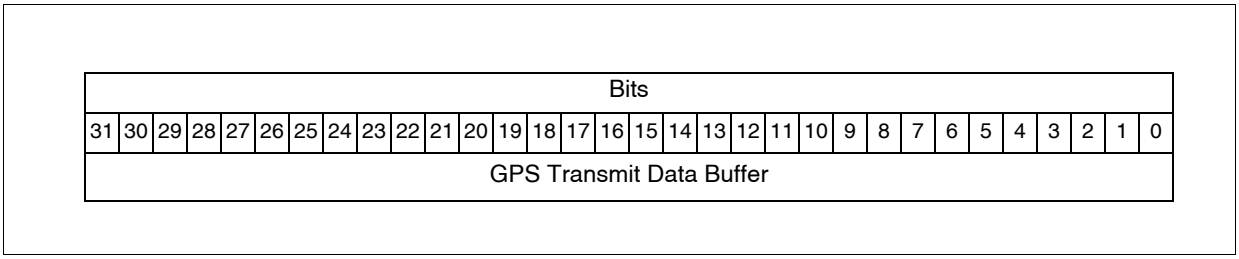


Figure B-30 RCIM III GPS Transmit Data Buffer

This is the GPS trasmit data buffer.

Offset: 04800 to 04FFF



C
Calculating RCIM Cable Propagation Delays

This appendix provides calculations to determine if your cable connections introduce signal delays.

Interconnect Details

The maximum cable length between each interconnected RCIM boards is 30 meters (~100 feet).

The clock runs at 400ns per tick. If the clock signal takes more than 400ns to make it to any given slave in the chain, clock skew will occur from that point on. The clock is re-driven by each pass-through slave.

Due to synchronization and re-drive of the serial data on the cable, each RCIM added to an RCIM chain adds about 200ns of delay in addition to approximately ~7ns for each meter of the cable, or ~200ns per 30-meter cable, for a total delay of about 400ns.

Two systems in an RCIM chain with the 30-meter cable will operate within a 400ns clock tick. Having more than two results in less precise synchronization.

Note that if a pass-through slave system is powered off, the cable clock will not be propagated to the slaves downstream from it. In this case, the downstream slaves will use their local oscillator instead of the cable clock.

The user just needs to be aware of the delays associated with each RCIM and cable in a chain to determine if the level of clock skew is acceptable for the application.