Real-Time Clock and Interrupt Module User's Guide

0891082-010
August 2001

Copyright 2001 by Concurrent Computer Corporation. All rights reserved. This publication or any part thereof is intended for use with Concurrent Computer Corporation products by Concurrent Computer Corporation personnel, customers, and end–users. It may not be reproduced in any form without the written permission of the publisher.

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

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

UNIX is a registered trademark of the Open Group.
Ethernet is a trademark of Xerox Corporation.
PowerMAX OS is a registered trademark of Concurrent Computer Corporation.
Power Hawk and PowerStack II/III are trademarks of Concurrent Computer Corporation.

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

Printed in U. S. A.

Revision History:	Level:	Effective With:
Original Issue -- July 1999	000	PowerMAX OS Release 4.3
Current Issue -- August 2001	010	PowerMAX OS Release 5.1

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). Refer to the following PowerMAX OS manuals for additional information on the RCIM. See the list of Related Publications below for additional manuals.

- *PowerMAX OS Real-Time Guide*
- *PowerMAX OS Guide to Real-Time Services*

Structure of Manual

This manual consists of a title page, this preface, a master table of contents, three chapters, and an index.

- Chapter 1, *Introduction*, contains a general overview of the RCIM.
- Chapter 2, *RCIM Hardware Considerations*, provides hardware preparation, installation instructions, RCIM hardware details and cabling information.
- Chapter 3, *RCIM Functional Description*, provides a functional description of the RCIM.

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 and 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 such option or arguments

Related Publications

Title	Pubs No.
Concurrent Computer Corporation Manuals:	
System Administration Manual (Volume 1)	0890429
System Administration Manual (Volume 2)	0890430
PowerMAX OS Real-Time Guide	0890466
PowerMAX OS Guide to Real-Time Services	0890479
Power Hawk Series 600 PowerMAX OS 'x.x' Release Notes (where 'x.x' = release number (reln))	0891058-reln
Power Hawk Series 700 PowerMAX OS 'x.x' Release Notes (where 'x.x' = release number (reln))	0891084-reln
Vendor Manuals - Motorola VMEbus Single Board Computer (SBC) Manuals: (See Note below)	
MVME2600 Series Single Board Computer Installation and Use Manual	V2600A/IH1
MVME4600 Series Single Board Computer Installation and Use	VMV4600A/IH1
PPCBug Firmware Package User's Manual (Parts 1)	PPCUGA1/UM
PPCBug Firmware Package User's Manual (Part 2)	PPCUGA2/UM
PPC1Bug Diagnostics Manual	PPCDIAA/UM
Note: The Motorola documents are available on the following web site at: PDF Library - **http://www.mcg.mot.com/literature**	
Vendor Manuals - Synergy VMEbus Single Board Computer (SBC) Manuals:	
VGM5 VMEbus Dual G3/G4 PowerPC Single Board Computer User Guide	98-0317/UG-VGM5-01
VSS4 Quad 750 PowerPC VMEbus Single Board Computer for DSP User Guide	99-0062/UG-VSS4-01
SMon PowerPC Series SBCs Developers Application & Debugger User Guide	99-0041/UG-PPSM-01

Related Specifications

Title	Pubs No.
IEEE - Common Mezzanine Card Specification (CMC)	P1386 Draft 2.0
IEEE - PCI Mezzanine Card Specification (CMC)	P1386.1 Draft 2.0
Compact PCI Specification	CPCI Rev 2.1 Dated 9/2/97

Contents

Illustrations

1
Introduction

1.1. Overview

The Real-Time Clock and Interrupt Module (RCIM) shown in Figure 1-1, is a multi-function PCI mezzanine card (PMC) designed for time-critical applications that require rapid response to external events, synchronized clocks and/or synchronized interrupts. The RCIM provides hardware support for features normally found only in symmetric multiprocessing systems.

The RCIM provides:

- synchronized clocks (tick timer and posix format clock)
- edge-triggered interrupts
- real-time clocks
- programmable interrupts

Some set of interrupts may be distributed and sent to all systems connected via a common RCIM chain. PowerMAX OS supports the RCIM on any supported Motorola or Synergy Single Board Computer (SBC) system.

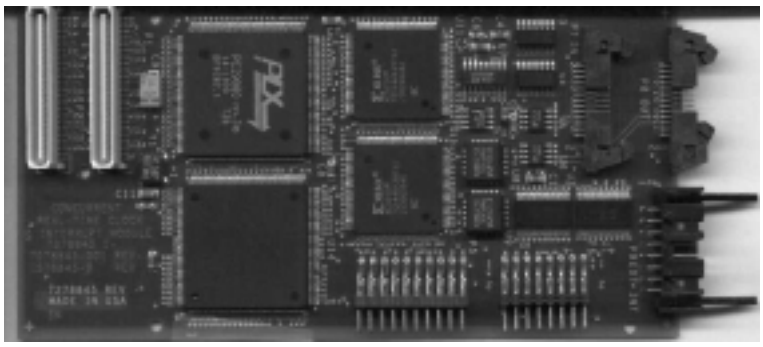

Figure 1-1. Real-Time Clock & Interrupt Module, Model CM4484

1.2. RCIM Specifications

Synchronized Clocks:	
- Posix	
Length	64 bits (two 32-bit words)
Resolution	High-order 32 bits - 1 second Low-order 32 bits
- Tick Timer	
Length	64 bits (two 32-bit words)
Resolution	64 bit counter of 400 ns ticks
Real-Time Clocks:	
Number	4
Length	32 bits
Resolution	1 microsecond (programmable to larger values)
Local Interrupts:	
External Edge-Triggered Interrupts	4
Real-Time Clocks	4
External Output Signals	4
Distributed Interrupts (8 total):	
Edge-Triggered Interrupt (ETI)	up to 4
Real-Time Clock (RTC)	up to 4
Input	software programmable: up to 4
Interrupt Response Time:	
Interrupt to user process	<8 microseconds
Packaging:	
Form Factor	IEEE P1386.1 PMC
Maximum cable length	16 ft.
External Connectors	16 position 0.1'' Latching Header
Environmental:	
Operating Temperature	10^{o} to 40^{o} C
Storage Temperature	-40^{o} to 65^{o} C
Relative Humidity	10 to 80% (non-condensing)
Power:	
Consumption	~5 watts

2
RCIM Hardware Considerations

2.1. Introduction

This chapter provides hardware preparation, installation instructions and information on the RCIM LED indicators and input/output connectors. Information on how to cable multiple RCIMs is also provided.

CAUTION

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

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

2.2. Unpacking Instructions

NOTE

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

Unpack 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 of the equipment.

2.3. Hardware Configuration

The RCIM mounts in a standard IEEE P1386 PCI mezzanine slot on a host SBC. On Synergy SBC boards, the RCIM board may also be mounted in a PEX PMC expansion board slot when a PEX board is installed. The RCIM includes a synchronization cable for daisy-chaining a master RCIM to one or more slave RCIMs. A connector is mounted on each RCIM for connection to external interrupts. Figure 1-1 illustrates the placement of connectors and LED indicators on the board.

Detailed information on the LEDs and connectors are provided in the following sections. Refer to "RCIM Cabling" on page 2-6 for information on connecting multiple RCIMs.

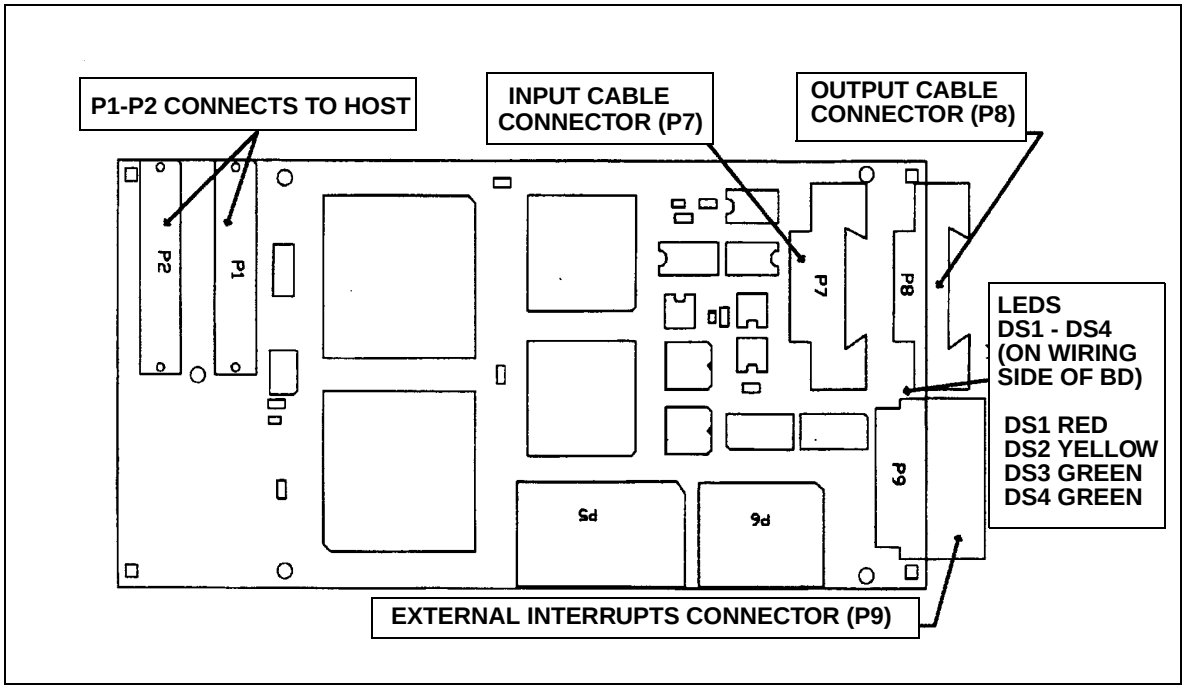

Figure 1-1. RCIM Input/Output Connector and LED Locations

There are four LEDs on the RCIM module. The function of each is described below.

LED	Number	Function
Red	DS1	If on after reset, indicates RCIM module failed power-up reset.
Yellow	DS2	If on, indicates cable clock not connected.
Green	DS3	If on, indicates activity in progress.
Green	DS4	If on, indicates power applied to RCIM.

2.3.1. Input Connector (P7)

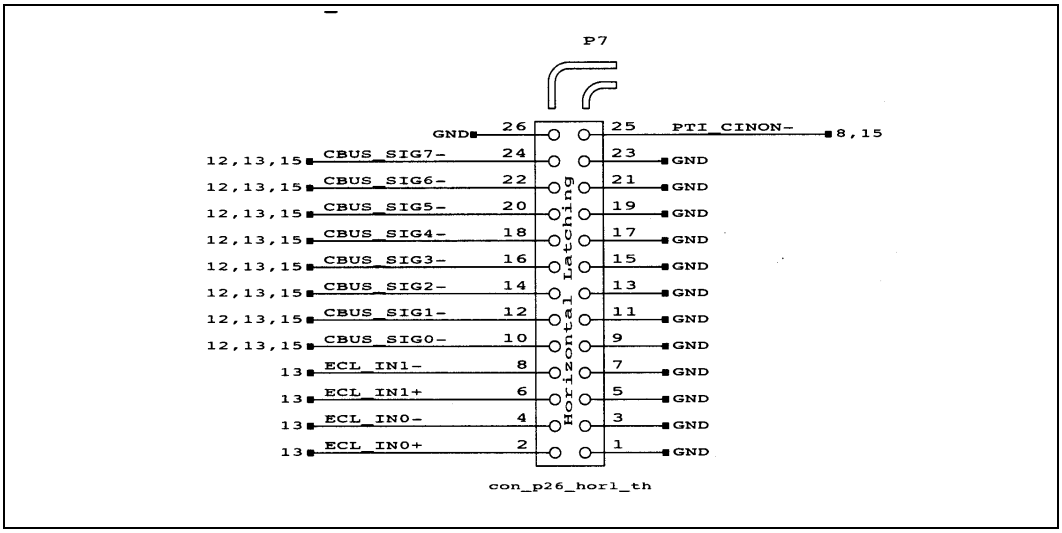

Figure 1-2. RCIM Input Cable Connector, P7

2.3.2. Output Connector (P8)

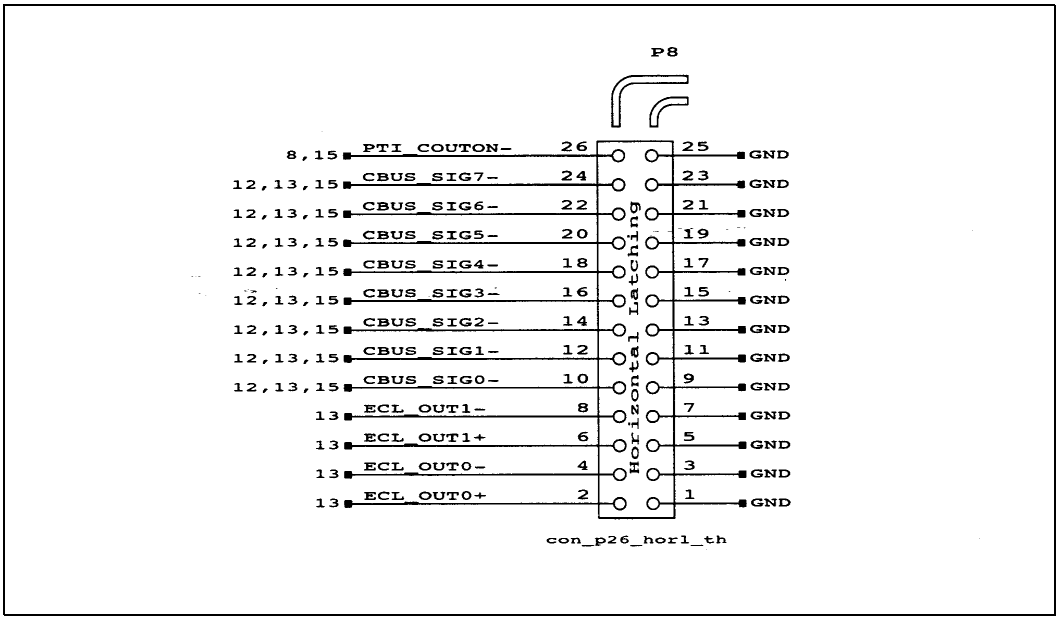

Figure 1-3. RCIM Output Cable Connector, P8

2.3.3. External Interrupts Connector (P9)

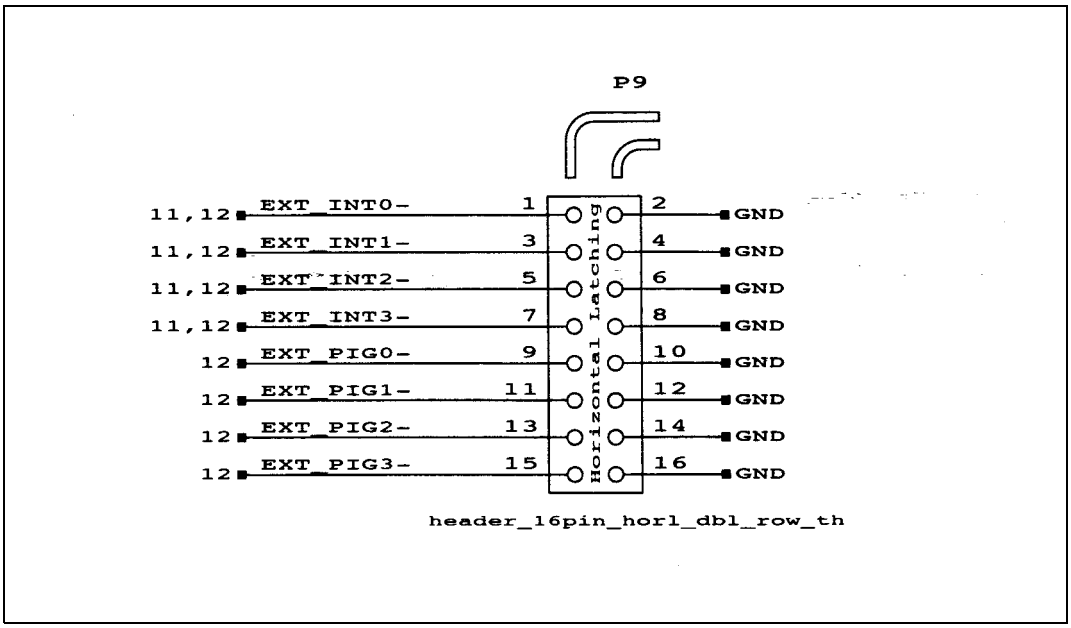

Figure 1-4. RCIM External Interrupt Connector, P9

2.4. Connection Modes

An RCIM may be used in one of three modes, depending on how it is cabled. Refer to Figure 1-5 for a functional block diagram view of this connection. Refer to section "RCIM Cabling" on page 2-6 for specific cabling details.

1. In isolated mode, there are no input/output connections to any other RCIMs.
2. In master mode, the RCIM is connected to one or more other RCIMs, and the RCIM is at the head of the chain. There is an output cable connection from a master RCIM but there is no input cable connection into it. The RCIM master is unique in that it controls the synchronized clocks.
3. In slave mode, the RCIM is connected to one or more other RCIMs, but is not at the head of the chain. There is an input cable connection into a slave RCIM and there may or may not be an output cable connection.

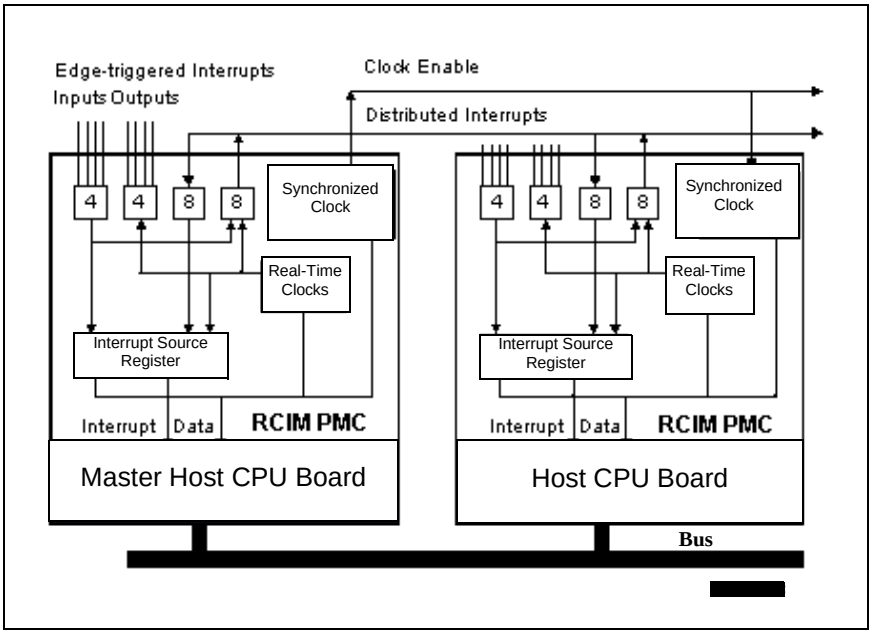

Figure 1-5. RCIM, Functional Connection Block Diagram

2.5. RCIM Cabling

Figure 1-6 illustrates the cabling for multiple RCIM connections. The RCIM includes a synchronization cable (PN 6010178-101 for daisy-chaining multiple RCIMs. Note that the RCIM board mounts with the component side facing the host CPU (wiring side out). The LEDS are visible from the wiring side. See LED placement in figure below.

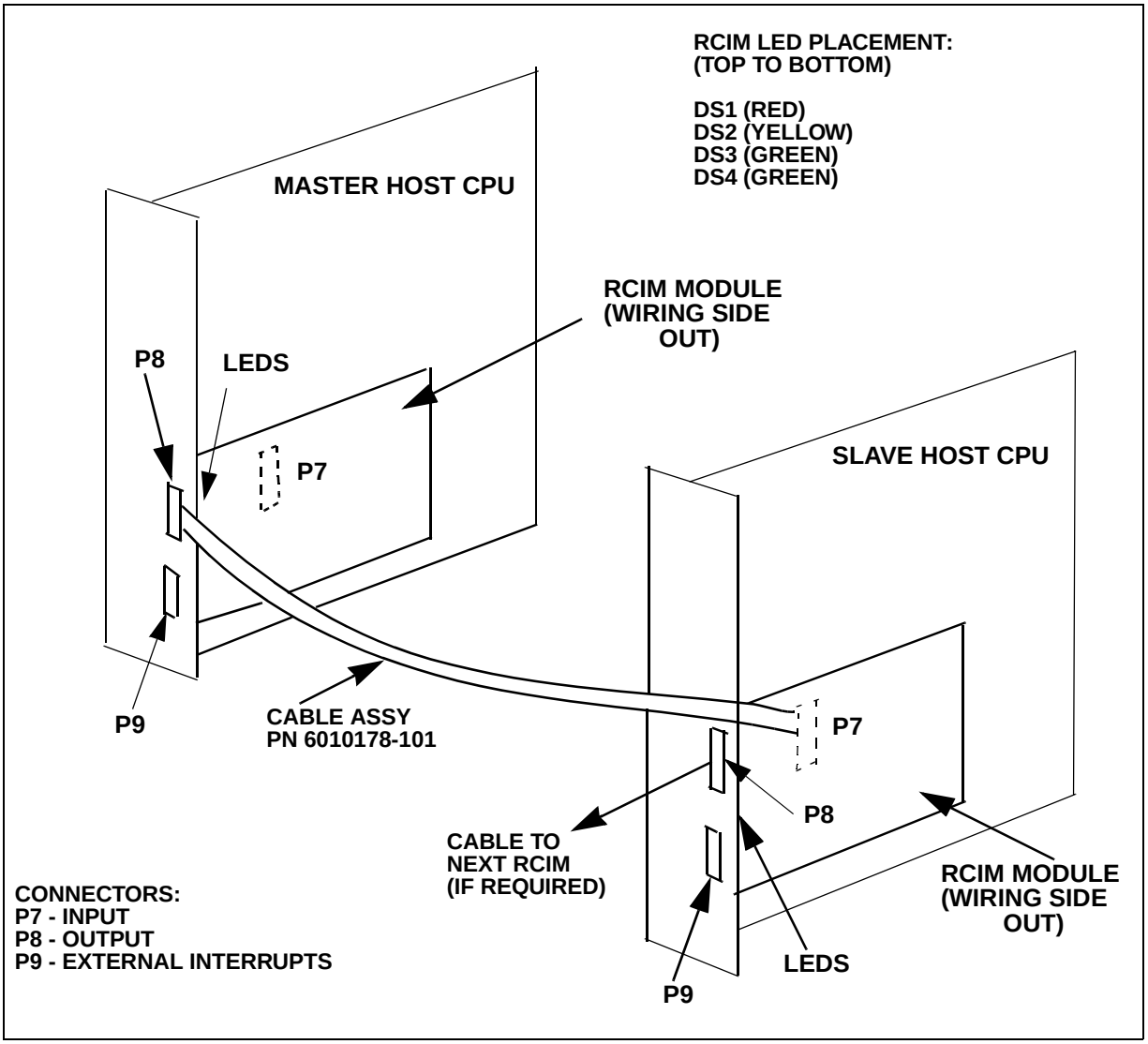

Figure 1-6. RCIM Cabling Diagram, Connector and LED Placements

3
Functional Description

3.1. Overview

This chapter provides a functional description of the following topics:

3.2. Synchronized Clocks

3.2.1. Overview of Synchronized Clocks

The RCIM provides two clocks. The primary clock is a 64-bit non-interrupting tick clock that counts each 400ns tick. The secondary 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.

The two RCIM clocks are independent of each other in the sense that their current “time” values will generally not agree. Also, when one clock is modified, the other is not affected.

Both the tick clock and POSIX clock “tick” in unison. Furthermore, when connected to other RCIMs, all clocks tick atomically.

The RCIM has a “synchronize” operation. When this function is performed, all tick clocks in an RCIM chain will be atomically reset to zero. (Synchronization will be covered in more detail later). As a result, once synchronized, the tick clocks provide a time base that is consistent for all connected SBCs.

It should be noted that unlike the standard processor interval timers, the RCIM tick clock will **NOT** contain an approximation of the time of day.

The "synchronize" operation does not affect the posix clocks. As a result, the POSIX clocks can generally not be used as a consistent time stamp on all connected systems.

The RCIM clocks can be read using **`clock_gettime(3c)`** or by direct access using **`mmap(2)`** with the device file **`/dev/sync_clock`**.

3.2.2. Synchronized Clock Library Routines

The RCIM clocks may be accessed using the following library routines (with a clock_id of CLOCK_SYNCHRONIZED for the tick clock and CLOCK_SYNCHRONIZED_POSIX for the POSIX clock):

`clock_gettime(3c)`	get current value of synchronized clock
`clock_getres(3c)`	get resolution of synchronized clock
`clock_synchronize(3c)`	initialize/synchronize synchronized clock
`clock_settime(3c)`	set value of synchronized clock

3.2.3. Direct Access to Synchronized Clocks

The device file **`/dev/sync_clock`** may also be used to access the RCIM clocks directly using **`mmap(2)`**. From the address returned by **`mmap(2)`**, 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/sys/rcim.h`** (with names beginning with RCIM_SYNCCLOCK_).

3.2.4. Synchronized Clock Tunable

The system tunable RCIM_CLOCK_NOTSYNC may be used to force clock independence, i.e. the clocks or the local RCIM will not be synchronized with other RCIM-connected SBCs. This tunable is only meaningful on a SBC system that contains an RCIM in slave mode. In this mode the clocks will tick independently of the other RCIMs and will not be affected when a synchronization operation is done.

The utility **`config(1m)`** should be used to examine and or modify this tunable.

3.2.5. Clock Synchronization

As briefly mentioned in the section “Overview of Synchronized Clocks”, the RCIM has a “synchronize” operation. This function is used to reset the tick counters on all connected RCIMs to a consistent value. The tick counters are reset to zero.

This “synchronize” operation will occur automatically in some situations. In some cases, it must be done manually using the **`clock_synchronize(1M)`** utility (refer to the next section).

Automatic clock synchronization (so that all RCIM tick counters have the same value) will occur in the following situations:

1. When the RCIM master system boots.
2. On closely-coupled systems when the target systems are auto-booted (i.e. the entire cluster of processors are booted as one).
3. When an RCIM slave system boots, if automatic clock synchronization has been configured by modifying the file **`/etc/idrc.d/rcim`** as indicated in that file.

   Note that this feature is disabled by default.

   Beware of the consequences if automatic clock synchronization is configured. The RCIM tick clock on all systems will be reset to zero. This may have an undesirable impact on any process using the clock during synchronization.

Manual clock synchronization is required whenever an RCIM slave system reboots after the RCIM master has booted (unless situations #2 or #3 described above apply).

3.2.6. clock_synchronize Utility

The **`clock_synchronize(1m)`** utility is used to perform various operations with the RCIM synchronized clocks.

With no options, this utility will synchronize the tick clock on all RCIM-connected SBCs. This operation can only be done on the RCIM master SBC system (refer to the previous section).

The “**`-i`**” option is used to run in interactive/diagnostic mode. This is used to start, stop the clocks, update their values as well as other operations. For more information, refer to the **`clock_synchronize(1m)`** system manual page.

The “**`-m`**” option is used to print the name of the SBC where the RCIM master is located (the system tunable RCIM_MASTER_NAME must be properly configured).

The “**`-s`**” option is used to print the RCIM connection mode.

Note that only the **root** user may use this utility.

3.2.7. Synchronizing POSIX Clocks

As mentioned, the RCIM POSIX clocks generally do **not** contain a consistent time value. However, this can be done manually using **`clock_synchronize(1m)`** in interactive mode. This procedure requires steps on all connected systems.

Manual synchronization is done by:

- disabling the RCIM master cable clock signal.
- updating all POSIX clocks on all systems to a consistent value.
- enabling the RCIM master cable clock signal.

This sequence is explained in more detail in the **`clock_synchronize(1m)`** system manual page.

Note that following this procedure, whenever a system reboots, its RCIM POSIX clock will no longer be synchronized with the POSIX clock on all other connected RCIMs.

3.3. Edge-Triggered Interrupts (ETIs)

3.3.1. Overview of ETIs

Each RCIM provides four edge-triggered interrupts (ETIs). Generally, ETIs are edge sensitive and will generate an interrupt when the leading edge of an input signal transitions from high to low or low to high. ETIs may also be level sensitive and will generate an interrupt when the input signal is high or low. The mode and polarity of each ETI may be specified using tunables.

3.3.2. ETI Device Files

The device files **`/dev/reti/etiN`** (where **N** = 0, 1, 2, 3) are used to access the four ETIs available on each SBC system configured with an RCIM.

3.3.3. Distributed ETIs

Any or all of the four ETIs on an RCIM may be configured to be distributed. A distributed ETI has its interrupts distributed to all SBCs connected by an RCIM chain.

The source of a distributed ETI may be located on any RCIM (RCIM master or RCIM slave(s)).

An ioctl (ETI_INFO) may be used to determine if a specified ETI is distributed, i.e. if its interrupts are sent to all connected SBCs. Refer to the **eti(7)** manual page for more information.

3.3.4. ETI Driver Functions

The following functions are available using **ioctl(2)** with an ETI device file:

ETI_ARM	arm the edge-triggered interrupt.
ETI_DISARM	disarm the edge-triggered interrupt.
ETI_ENABLE	enable the edge-triggered interrupt.
ETI_DISABLE	disable the edge-triggered interrupt.
ETI_REQUEST	generate software requested interrupt.
ETI_ATTACH_SIGNAL	request that a signal be generated when an edge-triggered interrupt is received.
ETI_VECTOR	get interrupt vector associated with ETI.
ETI_INFO	get information about ETI.

Note that the RCIM eti driver is compatible with the eti driver on Concurrent Night Hawk systems.

Refer to the **eti(7)** manual page for more information.

3.3.5. ETI Tunables

The following system tunables apply to RCIM edge-triggered interrupts:

RCIM_ETI_LEVEL	specifies for each eti, if it is edge-sensitive or level sensitive (by default, edge-sensitive).
RCIM_ETI_POLARITY	specifies for each eti, if it has positive or negative polarity (by default, negative).
RCIM_DISTRIB_INTR_[N]	may be used to configure a distributed ETI. Refer to “Distributed Interrupts” on page 3-9 for details.

The utility **config(1m**) should be used to examine and or modify these tunables.

3.4. Real-Time Clocks (RTCs)

3.4.1. Overview of RTCs

Each RCIM provides four real-time clocks (RTCs). Real-time clocks are used to generate an interrupt after a pre-loaded counter ticks down to zero.

The RCIM real-time clocks may be used in addition to, or instead of, the real-time clocks already available on the system.

3.4.2. RTC Device Files

The device files **`/dev/rrtc/2cN`** (where **N** = 0, 1, 2, 3) are used to access the four RTCs available on each system with an RCIM.

3.4.3. Distributed RTCs

Any or all of the four RTCs on an RCIM may be configured to be distributed. A distributed RTC has its interrupts distributed to all systems connected by an RCIM chain.

The source of a distributed RTC may be located on any RCIM (RCIM master or RCIM slave(s)).

An ioctl (RTCIOCINFO) may be used to determine if a specified RTC is distributed, i.e. if its interrupts are sent to all connected systems. Refer to the **`rtc(7)`** manual page for more information.

3.4.4. RTC Driver Functions

The following functions are available using **`ioctl(2)`** with an RTC device file:

RTCIOCSET	initialize RTC values
RTCIOCGET	retrieve RTC values
RTC_DIRECT	set RTC into direct programming mode
RTC_DEFAULT	set RTC to count down once
RTCIOCSETCNT	set RTC clock count
RTCIOCMODCNT	modify RTC clock count
RTCIOCGETCNT	get RTC clock count
RTCIOCRES	get RTC clock resolution
RTCIOCSTART	start RTC counting
RTCIOCSTOP	stop RTC counting
RTCIOCWAIT	block until RTC clock count reaches zero
RTCIOCFBS	set RTC clock mode for frequency-based scheduling (FBS)

IOCTLVECNUM	get interrupt vector for RTC
RTCIOCINFO	get information about RTC

Note that the rtc driver provides the same functions for an rtc on an RCIM as well as an rtc provided by the various system architecture supported by PowerMAX OS.

Refer to the **`rtc(7)`** manual page for more information.

3.4.5. RTC Tunables

The following system tunables apply to RCIM real-time clocks:

RCIM_DISTRIB_INTR_[N]	may be used to configure a distributed ETI. Refer to "Distributed Interrupts" on page 3-9 for more details.

The utility **`config(1m)`** should be used to examine and or modify this tunable.

3.5. External Output Signals

3.5.1. Overview of External Output Signals

Each RCIM provides four external output signals. These signals can be used as interrupt sources for other machines or used as signals to control external devices.

The external output signals can be driven from one of several different sources. The most common would be from the priority interrupt generators (PIGs). This provides full software control for generation of the output signals.

3.5.2. Attaching External Output Signals

An external output signal can be attached, using cabling, to trigger an edge-triggered interrupt on any system that contains an RCIM. Each edge-triggered interrupt is configured to be edge-sensitive or level sensitive and positive or negative polarity. For example, a positive polarity, edge-sensitive ETI will generate an interrupt when the signal transitions from negative to positive.

3.5.3. External Output Signal Tunables

There is one tunable associated with each of the four external output signals:

RCIM_EXT_OUTPUT_[N] (where N = 0, 1, 2 or 3)	specifies the source for external output signal N. The default is that PIG N is used to drive external output signal N. External output signals can also be driven by the output interrupt signal for any of the four edge-triggered interrupts, four real-time clocks or eight distributed interrupts.

The utility **`config(1m)`** should be used to examine and or modify these tunables.

3.6. Priority Interrupt Generators (PIGs)

3.6.1. Overview of PIGs

Each RCIM provides four (4) priority 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 force a distributed software interrupt to all SBCs connected via an RCIM chain.

3.6.2. PIG Device File

The device file **`/dev/pig`** may be 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)`**.

The pig register is a 32-bit register, one bit for each PIG. Pig 0 is controlled by bit 0, pig 1 by bit 1, pig 2 by bit 2 and pig 3 by bit 3. The remaining bits are unused.

Writing to the pig register, simply causes the output signal associated with the PIG to be positive or negative reflecting the inverter value in the pig register. Writing a 0 to a PIG register causes the output signal associated with the PIG to become positive. Conversely, writing a 1 to a PIG register causes the output signal associated with the PIG to become negative.

PIG value transitions must be at least 1.5 ns in duration in order to cause an interrupt to be generated.

Refer to the **`pig(7)`** manual page for more information.

3.6.3. Distributed PIGs

Any or all of the four PIGs on an RCIM may be configured to be distributed. A distributed PIG has its output interrupt signal distributed to all systems connected by an RCIM chain.

The source of a distributed PIG may be located on any RCIM (RCIM master or RCIM slave(s)).

3.6.4. PIG Tunables

The following system tunables apply to RCIM PIGs:

RCIM_DISTRIB_INTR_[N]	may be used to configure a distributed PIG. Refer to “Distributed Interrupts” (below).

The utility **`config(1m)`** should be used to examine and or modify this tunable.

3.7. Distributed Interrupts

3.7.1. Overview of Distributed Interrupts

Each RCIM may distribute interrupts to all SBCs connected via an RCIM chain. There are a total of eight distributed interrupts available. Distributed interrupts are like edge-triggered interrupts and must be armed and enabled for an interrupt to occur.

Any of the edge-triggered interrupts, real-time clocks or priority interrupt generators may be configured to be distributed. A distributed device file is associated with each of the eight distributed interrupts.

A distributed edge-triggered interrupt will generate simultaneously both its local edge-triggered interrupt and its associated distributed interrupt. If the edge-triggered interrupt is disarmed then neither interrupt will be generated. If the edge-triggered interrupt is armed but **not** enabled then neither interrupt will be generated until the ETI is enabled.

A distributed real-time clock will generate simultaneously both its local interrupt and its associated distributed interrupt.

A distributed priority interrupt generator will simultaneously send its signal locally and to its associated distributed interrupt.

3.7.2. Distributed Interrupts Device Files

The device files **`/dev/distrib_intrN`** (where N= 0, 1, 2, . . 7) are used to access the eight distributed interrupts available on each SBC with an RCIM.

3.7.3. Distributed Interrupt Driver Functions

A distributed interrupt is similar to an edge-triggered interrupt. Most of the functions available to edge-triggered interrupts are also available to distributed interrupts. The eti driver must be enabled to access distributed interrupts.

The following functions are available using **`ioctl(2)`** with a distributed interrupt device file:

DISTRIB_INT_ARM	arm the distributed interrupt.
DISTRIB_INT_DISARM	disarm the distributed interrupt.
DISTRIB_INT_ENABLE	enable the distributed interrupt.
DISTRIB_INT_DISABLE	disable the distributed interrupt.
DISTRIB_INT_ATTACH_SIGNAL	request that a signal be generated when an distributed interrupt is received.
DISTRIB_INT_VECTOR	get interrupt vector for distributed interrupt.
DISTRIB_INT_INFO	get information about distributed interrupt.

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

Refer to the **`distrib_intr(7)`** manual page for more information.

3.7.4. Distributed Device Interrupts

On the SBC where a distributed device resides, it may generate two interrupts: its local device interrupt (i.e. ETI 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, it will **not** generate an interrupt on the local system. By default, a distributed interrupt is in a disarmed state.

3.7.5. Configuring Distributed Interrupts

RCIM distributed interrupts must be configured on each SBC attached to the RCIM that is either broadcasting or receiving a distributed interrupt. For each of the eight distributed interrupt lines, the device that generates the interrupt must be specified. Any of the real-time clocks, edge-triggered interrupts or priority interrupt generators on any of the RCIM-attached SBCs could be the source of an interrupt that is distributed.

On a given SBC, only one device can drive a given distributed interrupt line. It is possible for more than one device to drive a given distributed interrupt line if those devices are on separate SBCs, though such configurations are of limited use. In this case, the distributed interrupt line will represent an OR of the interrupts generated by the multiple devices.

It is important that all RCIM-connected SBCs have the same configuration for the distributed interrupt lines of the RCIM. On the SBC where a local RCIM device is the source of a distributed interrupt, the device is associated with the correct distributed interrupt line and it is marked as "local". The "local" designation indicates that the device on the local RCIM will drive the distributed interrupt line. On other RCIM-connected SBCs, the distributed interrupt line must be associated with the same device type, but the line is specified as "remote". The "remote" designation indicates that this system will only receive the distributed interrupt. Note that on a remote system, it is unknown which foreign system is generating the interrupt.

By default, no distributed interrupts are configured on an SBC.

3.7.6. Distributed Interrupt Tunables

The following system tunables apply to RCIM distributed interrupts:

RCIM_DISTRIB_INTR_[N] (where N = 0, 1, 2. . . 7)	Specifies the source for distributed interrupt N. Distributed interrupts can be driven by ETIs, RTCs or PIGs. Also specifies if the device is local or remote.
RCIM_DISTRIB_LEVEL	Specifies for each distributed interrupt, if it is edge-sensitive or level sensitive (by default, edge-sensitive). Generally, this tunable should only be altered for distributed interrupts driven by priority interrupt generators (PIGs).
RCIM_DISTRIB_POLARITY	Specifies for each distributed interrupt, if it has positive or negative polarity (by default, negative). Generally, this tunable should only be altered for distributed interrupts driven by priority interrupt generators (PIGs).

The utility **`config(1m)`** should be used to examine and or modify these tunables.

3.7.7. Distributed Interrupt Information

There are two primary methods to get information about how distributed interrupts are configured. These methods are:

1. The DISTRIB_INT_INFO ioctl function applies to a distributed interrupt device will return information about a specific distributed interrupt. This information includes the source of the distributed interrupt (i.e., eti, rtc or pig) and also an indication of whether the source of the interrupt resides on the local system. Refer to the **`distrib_ intr(7)`** manual page for more information.

2. The RCIM_GET_INFO ioctl function for the device file **`/dev/rcim`** provides extended information on an RCIM including, connection mode, distributed interrupt configuration and tunable values. The header file **`/usr/include/sys/rcim.h`** describes the layout of the information returned. Refer to the **`rcim(7)`** manual page for more details.

   The utility **`rcimconfig(1m)`** described briefly in "rcimconfig Utility" below, displays the information returned from the RCIM_GET_INFO ioctl function using a very understandable format.

3.8. Additional RCIM Configuration Information

The following kernel modules must be built into a kernel in order to access a RCIM:

- rcim
- eti (unless ETIs and distributed interrupts are **not** used)
- rtc (unless RTCs are **not** used)

There are several system tunables used to configure the RCIM for usage appropriate for a given site. Most of these were described in earlier sections of this guide. An additional tunable not yet mentioned is described below.

RCIM_MASTER_NAME	specifies the hostname for the system that contains the RCIM master. This is used to access the system where the RCIM master is located.

3.9. rcimconfig Utility

The utility **`rcimconfig(1m)`** is used to view the configuration of an RCIM. It displays the following information:

- RCIM version
- connection mode and rcim master name
- configuration of ETIs
- configuration of RCIM RTCs
- configuration of external output signals
- configuration of PIGs
- configuration of distributed interrupts

Refer to the **`rcimconfig(1m)`** manual page for more details.

Index

O

P

R

S

T

U

Spine for 1/2” Binder

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up

PowerMAX OS

User

RCIM User’s Guide

0891082