KeyStone Architecture
Serial Rapid IO (SRIO)

User's Guide

Literature Number: SPRUGW1C
November 2010–Revised April 2019

Contents

List of Figures

List of Tables

Preface

About This Manual

The RapidIO peripheral used in KeyStone devices is called serial RapidIO (SRIO). This manual describes the general operation of SRIO, how this module is connected to the outside world, the features supported, SRIO registers, and examples of channel and queue operations.

[(1)]Notational Conventions

This document uses the following conventions:

- Commands and keywords are in **boldface** font.
- Arguments for which you supply values are in *italic* font.
- Terminal sessions and information the system displays are in screen font.
- Information you must enter is in **boldface screen font**.
- Elements in square brackets ([ ]) are optional.

Notes use the following conventions:

NOTE: Means reader take note. Notes contain helpful suggestions or references to material not covered in the publication.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

CAUTION

Indicates the possibility of service interruption if precautions are not taken.

WARNING

Indicates the possibility of damage to equipment if precautions are not taken.

Related Documentation from Texas Instruments

C66x CorePac User Guide	SPRUGW0
Enhanced Direct Memory Access 3 (EDMA3) for KeyStone Devices User Guide	SPRUGS5
General Purpose Input/Output (GPIO) for KeyStone Devices User Guide	SPRUGV1
Multicore Navigator for KeyStone Devices User Guide	SPRUGR9
Phase Locked Loop (PLL) Controller for KeyStone Devices User Guide	SPRUGV2
SerDes Implementation Guide for KeyStone I Devices	SPRABC1
SRIO Migration Guide (F to N) for KeyStone Devices	SPRABI0
SRIO Usage Considerations for KeyStone Devices	SPRABH4

(1) All trademarks are the property of their respective owners.

Introduction

The RapidIO peripheral used in KeyStone devices is called Serial RapidIO (SRIO). This chapter describes the general operation of a RapidIO system, how this module is connected to the outside world, the definitions of terms used within this document, and the features supported and not supported for SRIO.

Topic **Page**

1.1 General RapidIO System

RapidIO is a non-proprietary, high-bandwidth, system-level interconnect. It is a packet-switched interconnect intended primarily as an intrasystem interface for chip-to-chip and board-to-board communications at gigabyte-per-second performance levels. The architecture can be used in connected microprocessors, memory, and memory-mapped I/O devices that operate in networking equipment, memory subsystems, and general-purpose computing.

The principle features of RapidIO include:

- Flexible system architecture that allows peer-to-peer communication
- Robust communication with error-detection features
- Frequency and port-width scalability
- Operation that is not software-intensive
- High-bandwidth interconnect with low overhead
- Low pin count
- Low power
- Low latency

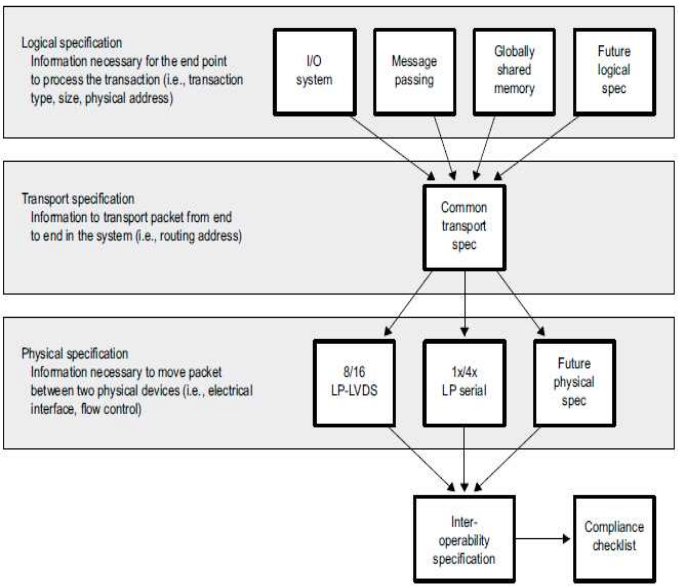
1.1.1 RapidIO Architectural Hierarchy

RapidIO is defined as a three-layer architectural hierarchy.

- **Logical layer:** Specifies the protocols, including packet formats, needed by endpoints to process transactions.
- **Transport layer:** Defines addressing schemes to correctly route information packets within a system.
- **Physical layer:** Contains the device-level interface information, such as the electrical characteristics, error management data, and basic flow control data.

In the RapidIO architecture, one specification for the transport layer is compatible with different specifications for the logical and physical layers.

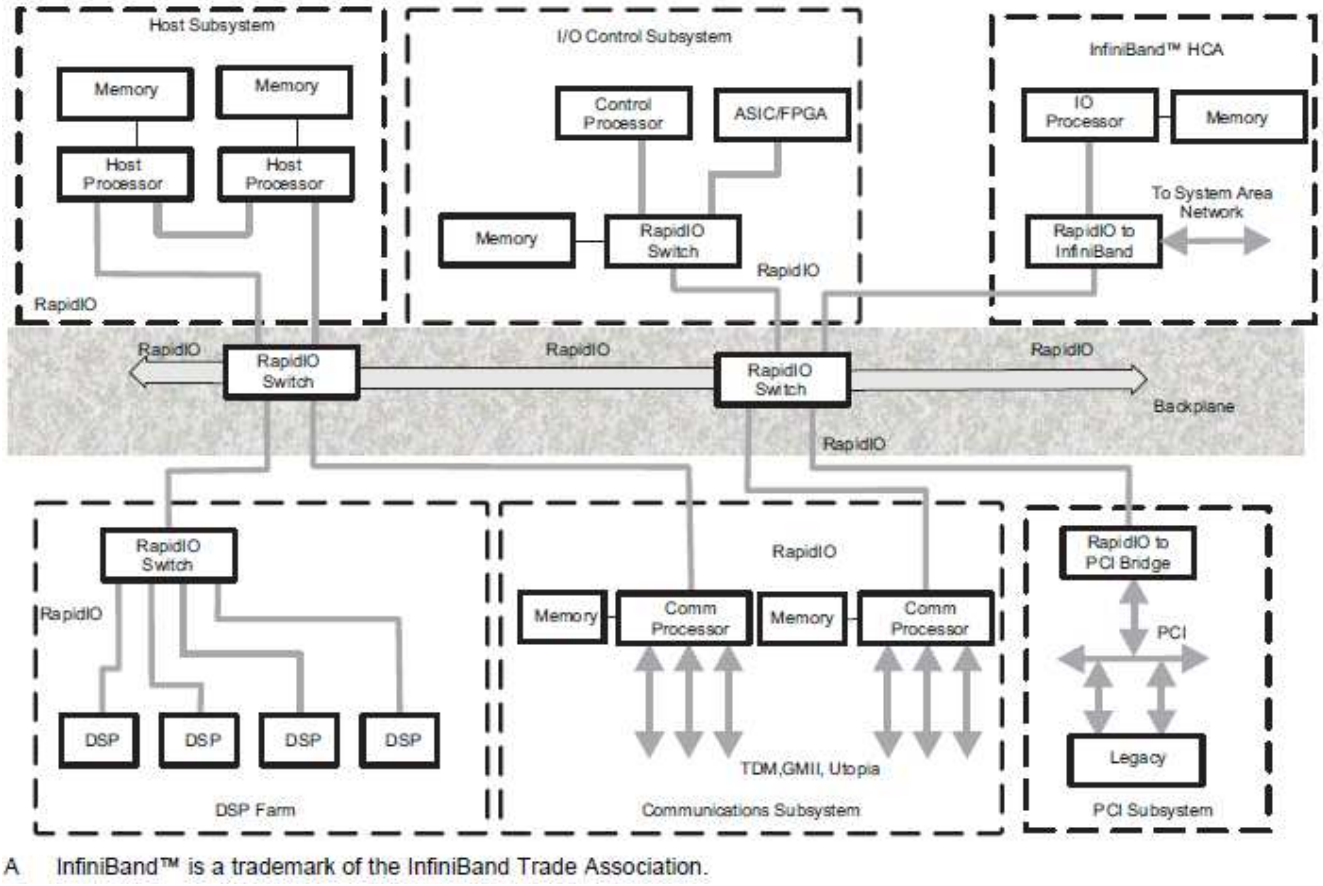
Figure 1-1. RapidIO Architectural Hierarchy



1.1.2 RapidIO Interconnect Architecture

The interconnect architecture is defined as a packet-switched protocol independent of a physical layer implementation. Figure 1-2 shows the interconnection system.

Figure 1-2. RapidIO Interconnect Architecture



A InfiniBand™ is a trademark of the InfiniBand Trade Association.

1.1.3 Physical Layer 1x/4x LP-Serial Specification

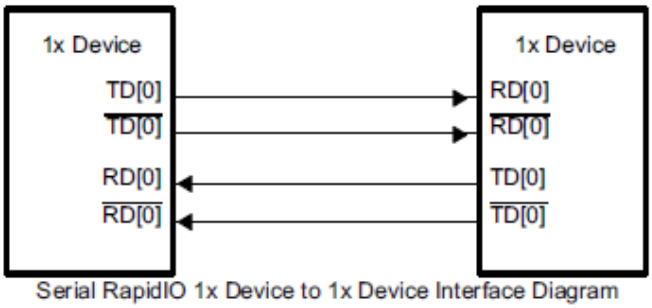
Currently, there are two physical layer specifications recognized by the RapidIO Trade Association: 8/16 LP-LVDS and 1x/4x LP-Serial. The 8/16 LP-LVDS specification is a point-to-point synchronous clock-sourcing DDR interface. The 1x/4x LP-Serial specification is a point-to-point, AC coupled, clock-recovery interface. The two physical layer specifications are not compatible.

SRIO complies with the 1x/4x LP-Serial specification. The serializer/deserializer (SerDes) technology in SRIO also aligns with that specification.

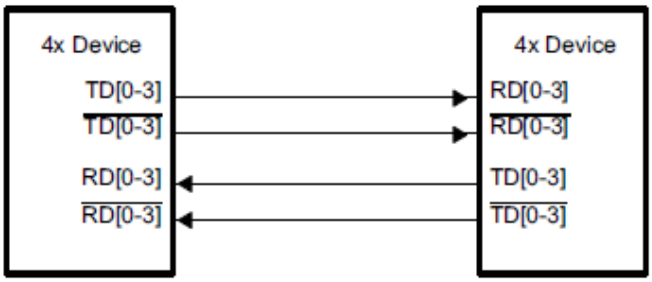
The RapidIO physical layer 1x/4x LP-Serial specification currently covers four frequency points: 1.25, 2.5, 3.125, and 5 Gbps. This defines the total bandwidth of each differential pair of I/O signals. An 8-bit/10-bit encoding scheme ensures ample data transitions for the clock-recovery circuits. Due to the 8-bit/10-bit encoding overhead, the effective data bandwidth per differential pair is 1.0, 2.0, 2.5, and 4 Gbps, respectively. Serial RapidIO only specifies these rates for both the 1x and 4x ports. A 1x port is defined as one TX and one RX differential pair. A 4x port is a combination of four of these pairs. This document describes a 4x RapidIO port that can also be configured as four 1x ports; this provides a scalable interface capable of supporting a data bandwidth of 1 to 16 Gbps.

Figure 1-3 shows how to interface two 1x devices and two 4x devices. Each positive transmit data line (TDx) on one device is connected to a positive receive data line (RDx) on the other device. Likewise, each negative transmit data line (TDx) is connected to a negative receive data line (RDx).

Figure 1-3. Serial RapidIO Device to Device Interface Diagrams



Serial RapidIO 1x Device to 1x Device Interface Diagram



Serial RapidIO 4x Device to 4x Device Interface Diagram

1.2 RapidIO Feature Support in SRIO

The following features are supported:

- RapidIO Interconnect Specification REV2.1.1-compliant
- LP-Serial Specification REV2.1.1-compliant
- 4X Serial RapidIO with auto-negotiation to
  - 1X port, optional operation for (4) 1X ports
  - 2X port, optional operation for (2) 2X ports
  - 2X port and 1X port operation, optional operation for (1) 2X port and (2) 1X ports
  - 4x port, operation for (1) 4x port
- Integrated clock recovery with TI SerDes
- Ability to run different ports at different baud rates (only integer multiple rates supported: 2.5G and 5G supported, 3.125G and 5G are not supported)
- Hardware error handling, including CRC
- Differential CML signaling supporting AC and DC coupling
- Support for 1.25, 2.5, 3.125, and 5 Gbps rates
- Powerdown option for unused ports
- Read, write, write with response, streaming write, out-going Atomic, maintenance operations
- Interrupt generation to the CPU (Doorbell packets and Internal scheduling)
- Support for 8b and 16b device ID
- Support for receiving 34b addresses
- Support for generating 34b, 50b, and 66b addresses
- Support for data sizes: byte, half-word, word, double-word
- Defined as Big Endian
- Direct IO transfers
- Message passing transfers
- Data payloads to 256B

- Single message generation up to 16 packets
- Elastic Store FIFO for clock domain handoff
- Short Run- and Long Run-compliant
- Support for error management extensions
- Support for congestion control extensions
- Support for multicast ID
- Short and long control symbols supported
- Support for IDLE1 sequences with a maximum baud rate of 5 Gbps
- Strict priority segment interleaving across protocol units based on priority and CRF

The following features are not supported:

- Compliance with the Global Shared Memory specification (GSM)
- 8/16 LP-LVDS compatible
- Destination support of RapidIO atomic operations

1.3 Standards

The RIO peripheral complies with REV 2.1.1 of the RapidIO Interconnect specification and REV 2.1.1 of the LP-Serial specification. The serial RapidIO AC specification contains two classes of drivers, designated as *long run* and *short run*. The long run specification applies to long backplane applications with a minimum of 50-cm traces and two or more connectors. The short run specification is designed for low-power applications. It is typically used for links on the same board or short backplane connections. The difference between the two classes is the driver's Vod.

Non-supported features are highlighted in this document. The GSM logical layer extension, which supports coherent externally shared memory access, is not supported at this time. This is an optional and separate specification.

Target atomic operations, including increment, decrement, test-and-swap, set, and clear are not supported for internal L2 memory or registers. Atomic request operations to external devices are supported.

1.4 External Devices Requirements

SRIO provides a seamless interface to all devices that are compliant with V1.2 of the RapidIO *Physical Layer 1x/4x LP-Serial Specification*. This includes ASIC, microprocessor, DSP, and switch fabric devices from multiple vendors. Compliance to the specification can be verified with bus-functional models available through the RapidIO Trade Association, as well as test suites currently available for licensing.

SRIO Functional Description

Topic **Page**

2.1 Overview

2.1.1 Peripheral Data Flow

This peripheral is an externally-driven slave module capable of acting as a master within the DSP chip. This means that an external device can push (burst write) data to the DSP as needed, without having to generate an interrupt to the CPU or without relying on the DSP EDMA. This has several benefits. It cuts down on the total number of interrupts, reduces handshaking (latency) associated with read-only peripherals, and frees up the EDMA for other tasks.

SRIO specifies data packets with payloads up to 256 bytes. Many times, transactions span multiple packets. RapidIO specifies a maximum of 16 packets per message. Although a request is generated for each packet transaction so that the DMA can transfer the data to L2 memory, an interrupt is generated only after the final packet of the message. This interrupt notifies the CPU that data is available in L2 memory for processing.

As an endpoint device, the peripheral accepts packets based on the destination ID. There are two mode-selectable options for packet acceptance. The first option accepts only packets whose DestIDs match the local DeviceID. This provides a level of security. The second option is system multicast operation. When multicast is enabled by Table 2-26, all incoming packets matching the deviceIDs mentioned in the note below and the multicast deviceIDs registers shown in Table 2-1 are accepted.

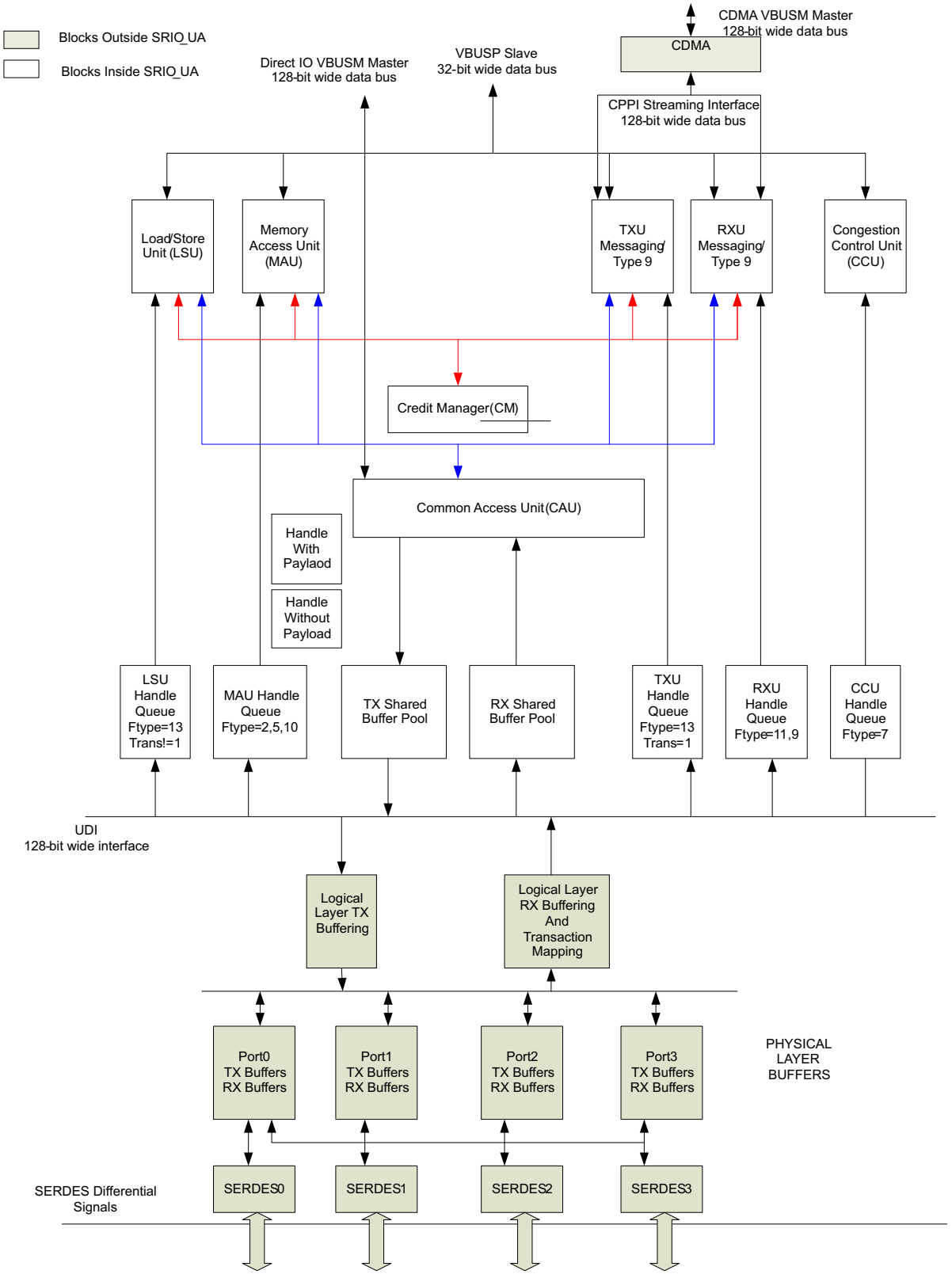
NOTE: DeviceID Register 1 through DeviceID Register 15 inherit their values from the Base Routing Registers (TLM port n BRR pattern and match registers— see Section 3.18.27).

Table 2-1. Registers Checked for DeviceID

DeviceID Type	Register Name	Address Offset
Multicast ID	RapidIO Multicast ID0	0x00C0
	RapidIO Multicast ID1	0x00C4
	RapidIO Multicast ID2	0x00C8
	RapidIO Multicast ID3	0x00CC
	RapidIO Multicast ID4	0x00D0
	RapidIO Multicast ID5	0x00D4
	RapidIO Multicast ID6	0x00D8
	RapidIO Multicast ID7	0x00DC

Data flow through the peripheral can be explained using the high-level block diagram shown in Figure 2-1.

Figure 2-1. Peripheral Module



High-speed data enters from the device pins into the RX block of the SerDes macro. The RX block is a differential receiver that expects a minimum of 175-mV peak-to-peak differential input voltage (Vid). Level shifting is performed in the RX block, such that the output is single-ended CMOS. The serial data is then fed to the SerDes clock recovery block. The sole purpose of this block is to extract a clock signal from the data stream. To do this, a low-frequency reference clock is required, $1/10^{th}$ or $1/20^{th}$ the data rate. For example, for 3.125-Gbps data, a refclk of 312.5 Mhz, 156.25 Mhz, or 125 MHz is needed. Typically, this clock comes from an off-chip stable crystal oscillator and is a LVDS device input separate to the SerDes. This clock is distributed to the SerDes PLL block, which multiplies that frequency up to that of the data rate. Multiple high-speed clock phases are created and routed to the clock recovery blocks. The clock recovery blocks further interpolate between these clocks to provide maximum unit interval (UI) resolution on the recovered clock. The clock recovery block samples the incoming data and monitors the relative positions of the data edges. With this information, it can provide the data and a center-aligned clock to the S2P block. The S2P block uses the newly recovered clock to demux the data into 10-bit or 20-bit words. At this point, the data leaves the SerDes macro at $1/20^{th}$ the pin data rate, accompanied by an aligned byte clock. 5G operation is supported using a 20-bit SerDes module.

Within the physical layer, the data next goes to the 8b/10b decode block. 8b/10b encoding is used by RapidIO to ensure adequate data transitions for the clock recovery circuits. Here the 20 percent encoding overhead is removed as the 10-bit data is decoded to the raw 8-bit data. At this point, the recovered byte clock is still being used.

The next step is clock synchronization and data alignment. These functions are handled by the FIFO and lane de-skewing blocks. The FIFO provides an elastic store mechanism used to handoff between the recovered clock domains and a common system clock. After the FIFO, the four lanes are synchronized in frequency and phase, whether 1X, 2X, or 4X mode is being used. The FIFO is 8 words deep. The lane de-skew is only meaningful in the 4X mode, where it is used to align each channel's word boundaries, such that the resulting 32-bit word is correctly aligned.

The CRC error detection block keeps a running tally of the incoming data and computes the expected CRC value for the 1X, 2X or 4X mode. The expected value is compared against the CRC value at the end of the received packet.

After the packet reaches the logical layer, the packet fields are decoded and payload is buffered. Depending on the type of received packet, the packet routing is handled by functional blocks which control the DMA access. Figure 2-1 shows these blocks.

The load/store unit (LSU) controls the transmission of direct I/O packets, and the memory access unit (MAU) controls the reception of direct I/O packets. The LSU also controls the transmission of maintenance packets. Message packets are transmitted by the TXU and received by the RXU. These four units use the internal DMA to communicate with internal memory, and they use buffers and receive/transmit ports to communicate with external devices.

2.1.2 Clock Summary

Figure 2-2. Clock Diagram

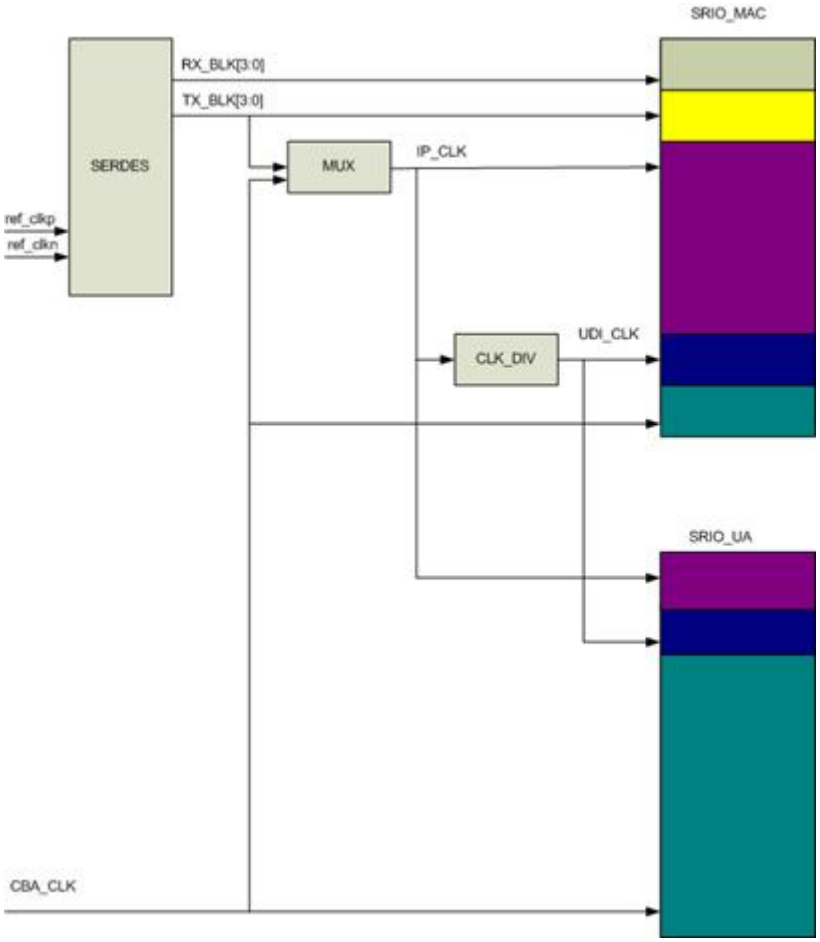


Table 2-2. Clock Summary

Clock Name	Frequency	Description
Refp_clk/Refn_clk	SERDES spec – Multiple possible frequencies available	Differential Input reference clk
cba_clk	1/3 CPU clock frequency	Clock used by majority of the logical layer and DMA
Txbclk[3:0]	1/20th the baud rate, i.e. 3.125-Gbaud data rate means a 156.25-Mhz Txbclk	TX byte clock from the SERDES used to send data. All lanes do not have to run at the same frequency; however, due to the Serdes restriction they must be exact multiples of each other.
Rxbclk[3:0]	1/20th the baud rate, i.e. 3.125-Gbaud data rate means a 156.25-Mhz Txbclk	They are the recovered RX clock from the SERDES. All lanes do not have to run at the same frequency; however, due to the Serdes restriction they must be exact multiples of each other. They must match the frequency of the TXbclk
Ip_clk	Highest frequency of the txbclks or the CBA_CLK (provided cba_clk is faster than the txbclks).	This is the output of a clockmux, selecting the fastest of the txbclks or additionally the cba_clk, provided cba_clk is faster than the txbclks. This clock is gated off if the peripheral is disabled. This clock is selected based on the programming of SYS_CLK_SEL and SYS_CLK_VBUSP in the RIO_PER_SET_CNTL1 (Address offset 0x0018)
udi_clk	Ip_clk/2	This is half the frequency of the ip_clk.

2.1.3 SRIO Packets

The RapidIO data stream consists of data fields pertaining to the logical layer, the transport layer, and the physical layer.

- The logical layer consists of the header (defining the type of access) and the payload (if present).
- The transport layer is somewhat dependent on the physical topology in the system, and consists of source and destination IDs for the sending and receiving devices.
- The physical layer is dependent on the physical interface (such as serial versus parallel RapidIO) and includes priority, acknowledgment, and error-checking fields.

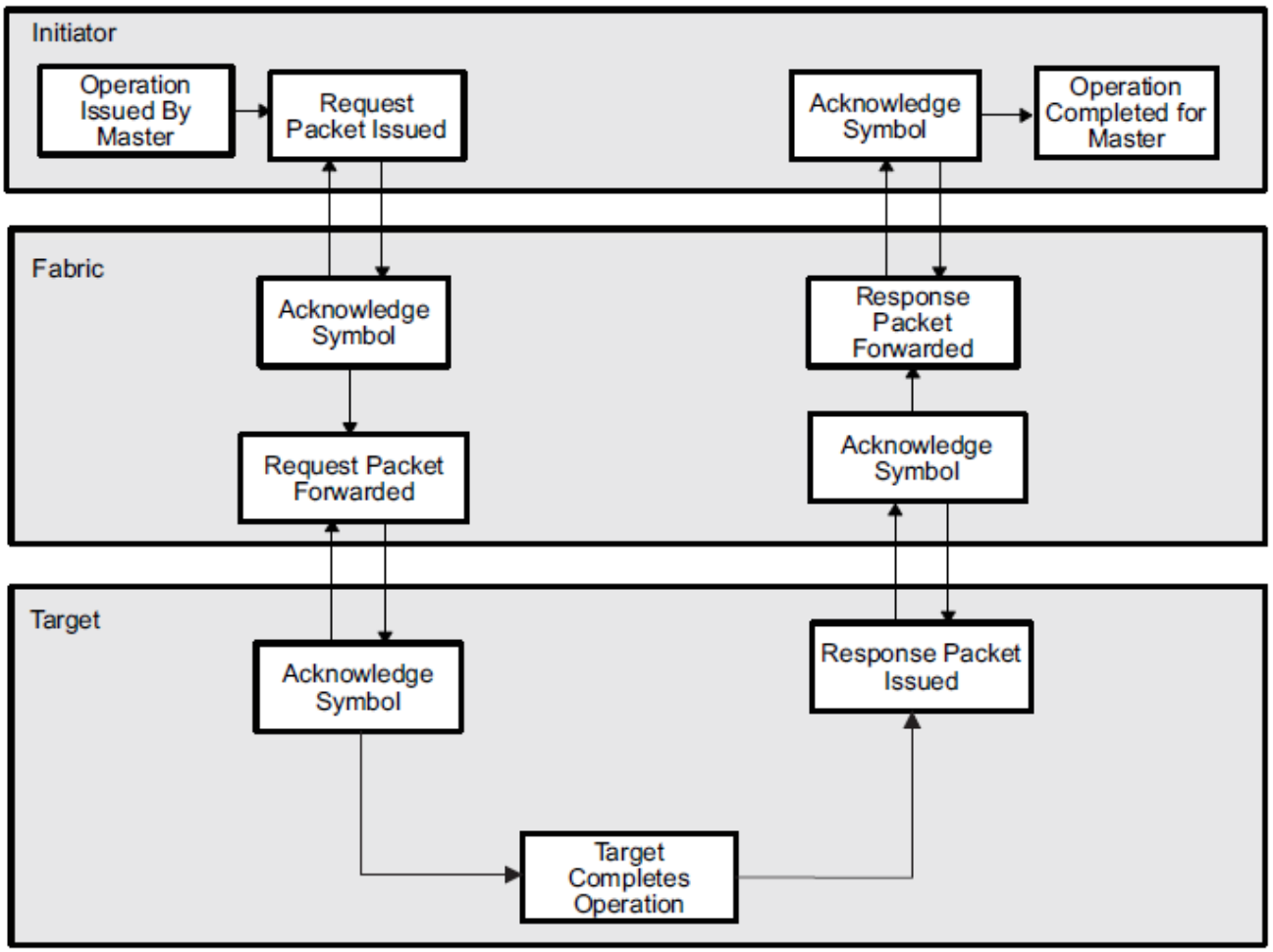
2.1.3.1 Operation Sequence

SRIO transactions are based on request and response packets. Packets are the communication element between endpoint devices in the system. A master or initiator generates a request packet which is transmitted to a target. The target then generates a response packet back to the initiator to complete the transaction.

SRIO endpoints are typically not connected directly to each other but instead have intervening connection fabric devices. Control symbols are used to manage the flow of transactions in the SRIO physical interconnect. Control symbols are used for packet acknowledgment, flow control information, and maintenance functions.

Figure 2-3 shows how a packet progresses through the system.

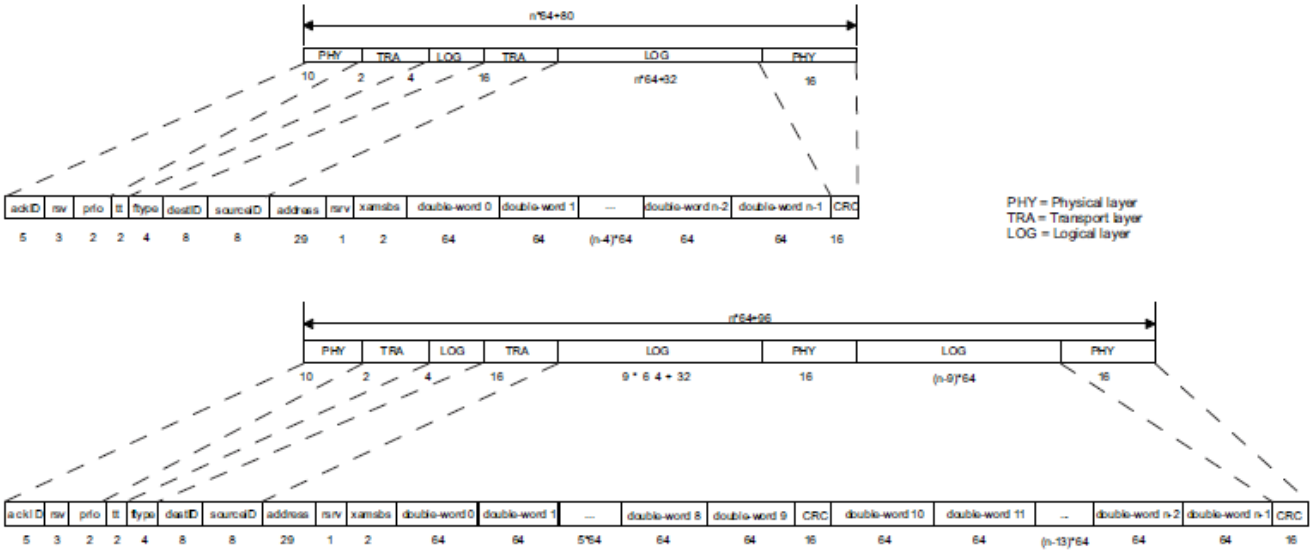
Figure 2-3. Operation Sequence



2.1.3.2 Example Packet—Streaming Write

Figure 2-4 shows an example packet as two data streams. The first is for payload sizes of 80 bytes or less, while the second applies to payload sizes of 80 to 256 bytes. SRIO packets must have a length that is an even integer of 32 bits. If the combination of physical, logical and transport layers has a length that is an integer of 16 bits, a 16-bit pad of value 0000h is added to the end of the packet, after the CRC (not shown). Bit fields that are defined as reserved are assigned to logic 0s when generated and ignored when received. All request and response packet formats are described in the RapidIO *Input/Output Logical Specification* and *Message Passing Logical Specification*.

Figure 2-4. 1X/RX RapidIO Packet Data Stream (Streaming-Write Class)



NOTE: Figure 2-4 assumes that addresses are 32-bit and device IDs are 8-bit.

The device ID, being an 8-bit field, addresses up to 256 nodes in the system. If 16-bit addresses were used, the system could accommodate up to 64k nodes.

The data stream includes a cyclic redundancy code (CRC) field to ensure the data was received correctly. The CRC value protects the entire packet except the ackID and one bit of the reserved PHY field. The peripheral checks the CRC automatically in hardware. If the CRC is correct, a packet-accepted control symbol is sent by the receiving device. If the CRC is incorrect, a packet-not-accepted control symbol is sent so that transmission can be retried.

2.1.3.3 Control Symbols

Control symbols are physical-layer message elements that manage link maintenance, packet delimiting, packet acknowledgment, error reporting, and error recovery. All transmitted data packets are delimited by start-of-packet and end-of-packet delimiters. SRIO control symbols are 24 bits long and are protected by their own CRC (see Figure 2-5). Control symbols provide two functions: stype0 symbols convey the status of the port transmitting the symbol, and stype1 symbols are requests to the receiving port or transmission delimiters. They have the following format, detailed in Section 3 of the RapidIO *Physical Layer 1x/4x LP-Serial Specification*.

Figure 2-5. Control Symbol Format

Delimiter	First Byte		Second Byte		Third Byte	
Sc or PD	stype0	Parameter0	parameter1	stype1	cmd	CRC
8	3	5	5	3	3	5

Control symbols are delimited by special characters at the beginning of the symbol. If the control symbol contains a packet delimiter (start-of-packet, end-of-packet, and so forth), the special character PD (K28.3) is used. If the control symbol does not contain a packet delimiter, the special character SC (K28.0) is used. This use of special characters provides an early warning of the contents of the control symbol. The CRC does not protect the special characters, but an illegal or invalid character is recognized and flagged as packet-not-accepted. Since control symbols are known length, and do not need end-delimiters.

The type of received packet determines how the packet routing is handled. Reserved or undefined packet types are destroyed before being processed by the logical layer functional blocks. This prevents erroneous allocation of resources. Unsupported packet types receive an error response packet.

2.1.3.4 SRIO Packet Type

The type of SRIO packet is determined by the combination of Ftype and Ttype fields in the packet. Table 2-3 lists all supported combinations of Ftype/Ttype and the corresponding decoded actions on the packets.

Table 2-3. Packet Types

FType	TType (4'b)	Operation
0	don't care	None
1	don't care	None
2	0100	NREAD
	1100	Atomic increment
	1101	Atomic decrement
	1110	Atomic set
	1111	Atomic clear
	Others	
3	don't care	None
4	don't care	None
5	4'b0100	NWRITE
	4'b0101	NWRITE_R
	4'b1110	Atomic test & swap
	Others	
6	don't care	SWRITE
7	don't care	Congestion control
8	4'b0000	Maintenance Read
	4'b0001	Maintenance Write
	4'b0010	Maintenance Read Response
	4'b0011	Maintenance Write Response
	4'b0100	Maintenance Port-write
	Others	
9	don't care	Data Streaming (CPDMA packet type 30)
10	don't care	Doorbell
11	don't care	Message (CPDMA packet type 31)
12	don't care	None
13	4'b0000	Response (+Doorbell Response)
	4'b0001	Message Response
	4'b1000	Response w/payload
	other	
14	don't care	None
15	don't care	None

2.2 SRIO Pins

The SRIO device pins are high-speed differential signals based on current-mode logic (CML) switching levels. The transmit and receive buffers are self-contained within the clock-recovery blocks. The reference clock input is not incorporated into the SerDes macro. It uses a differential input buffer that is compatible with the LVDS and LVPECL interfaces available from crystal oscillator manufacturers. Table 2-4 describes the device pins for the SRIO peripheral.

Table 2-4. Pin Description

Pin Name	Pin Count	Signal Direction	Description
RIOTX3/ $\overline{\text{RIOTX3}}$	2	Output	Transmit Data - Differential point-to-point unidirectional bus. Transmits packet data to a receiving device's RX pins. Most significant bits in 1 port 4X device. Used in 4 port 1X device.
RIOTX2 $\overline{\text{RIOTX2}}$	2	Output	Transmit Data - Differential point-to-point unidirectional bus. Transmits packet data to a receiving device's RX pins. Bit used in 4 port 1X device and in 1 port 4X device.
RIOTX1/ $\overline{\text{RIOTX1}}$	2	Output	Transmit Data - Differential point-to-point unidirectional bus. Transmits packet data to a receiving device's RX pins. Bit used in 4 port 1X device and in 1 port 4X device.
RIOTX0/ $\overline{\text{RIOTX0}}$	2	Output	Transmit Data - Differential point-to-point unidirectional bus. Transmits packet data to a receiving device's RX pins. Bit used in 1 port 1X device, 4 port 1X device, and 1 port 4X device.
RIORX3/ $\overline{\text{RIORX3}}$	2	Input	Receive Data - Differential point-to-point unidirectional bus. Receives packet data for a transmitting device's TX pins. Most significant bits in 1 port 4X device. Used in 4 port 1X device.
RIORX2 $\overline{\text{RIORX2}}$	2	Input	Receive Data - Differential point-to-point unidirectional bus. Receives packet data for a transmitting device's TX pins. Bit used in 4 port 1X device and in 1 port 4X device.
RIORX1/ $\overline{\text{RIORX1}}$	2	Input	Receive Data - Differential point-to-point unidirectional bus. Receives packet data for a transmitting device's TX pins. Bit used in 4 port 1X device and in 1 port 4X device.
RIORX0/ $\overline{\text{RIORX0}}$	2	Input	Receive Data - Differential point-to-point unidirectional bus. Receives packet data for a transmitting device's RX pins. Bit used in 1 port 1X device, 4 port 1X device, and 1 port 4X device.
RIOCLK/ $\overline{\text{RIOCLK}}$	2	Input	Reference Clock Input Buffer for peripheral clock recovery circuitry.

2.3 Functional Operation

This section describes the functional operation of the SRIO block.

2.3.1 SerDes Macro and its Configurations

SRIO offers many benefits by allowing a scalable non-proprietary interface. With the use of TI's SerDes macros, the peripheral is adaptable and bandwidth scalable. The same peripheral can be used for all four frequency nodes specified in V1.2 of the *RapidIO Interconnect Specification* (1.25, 2.5, 3.125 and 5 Gbps). This allows the user to design to only one protocol throughout the system and selectively choose the bandwidth, eliminating the need for user-proprietary protocols in many instances, and provides a faster design turn and production ramp. Because this interface is serial, the application space is not limited to a single board. It propagates into backplane applications as well. Integration of these macros on an ASIC or DSP allows you to reduce the number of discrete components on the board and eliminates the need for bus driver chips.

There are some valuable additional features built into the TI SerDes macro. System optimization can be managed to meet individual customer applications. For example, control registers within the SerDes allow you to adjust the TX differential output voltage (Vod) on a per-driver basis. This allows power savings on short trace links (on the same board) by reducing the TX swing. Similarly, data edge rates can be adjusted through the control registers to help reduce any EMI affects. Unused links can be powered down individually without affecting the working links.

The SerDes macro is a self-contained macro that includes transmitter (TX), receiver (RX), phase-locked-loop (PLL), clock recovery, serial-to-parallel (S2P), and parallel-to-serial (P2S) blocks. The internal PLL multiplies a user-supplied reference clock. All loop-filter components of the PLL are on-chip. Likewise, the differential TX and RX buffers contain on-chip termination resistors. The only off-chip component requirement is for DC-blocking capacitors.

2.3.1.1 Enabling the PLL

The physical layer SerDes has a built-in PLL, used for the clock-recovery circuitry. The PLL is responsible for clock multiplication of a slow-speed reference clock. This reference clock has no timing relationship to the serial data and is asynchronous to any CPU system clock. The multiplied high-speed clock is routed only within the SerDes block; it is not distributed to the remaining blocks of the peripheral, nor is it a boundary signal to the core of the device. It is important to have a good quality reference clock and to isolate it and the PLL from all noise sources.

The SerDes macro is configured with the registers SRIO_SERDES_CFGPLL, SRIO_SERDES_CFGRX[3-0], SRIO_SERDES_CFGTX[3-0], and SRIO_SERDES_RSVD. To enable the internal PLL, the ENPLL bit of SRIO_SERDES_CFGPLL must be set. After setting this bit, it is necessary to allow 1s for the regulator to stabilize. Thereafter, the PLL takes no longer than 200 reference clock cycles to lock to the required frequency, provided RIOCLK and RIOCLK are stable.

To ensure that the PLL has stabilized, the lock bit of the SRIO_SERDES_STS register (address location 0x02620154) can be polled to determine the PLL state. For more information on this register, refer to *Section 3.2.1*.

2.3.1.2 Enabling the Receiver

To enable a receiver for deserialization, the ENRX bit of the associated SERDES_CFGRX *n*_CNTL registers must be set high. The fields of SERDES_CFGRX *n*_CNTL are shown in Figure 3-4 and Table 3-12.

When ENRX is low, all digital circuitry within the receiver is disabled, and clocks gated off. All current sources within the receiver are fully powered down, with the exception of those associated with the loss of signal detector and IEEE1149.6 boundary scan comparators. Loss of signal powerdown is independently controlled through the LOS bits of SERDES_CFGRX *n*_CNTL. When enabled, the differential signal amplitude of the received signal is monitored. Whenever loss of signal is detected, the clock recovery algorithm is frozen to prevent the phase and frequency of the recovered clock from being modified by low-level signal noise.

The clock recovery algorithms listed in the CDR bits operate to adjust the clocks used to sample the received message, so that the data samples are taken midway between data transitions. The second-order algorithm can be optionally disabled, and both can be configured to optimize their dynamics. Both algorithms use the same basic technique for determining whether the sampling clock is ideally placed, and if not, whether it needs to be moved earlier or later. When two contiguous data samples are different, the phase sample between the two is examined. Eight data samples and nine phase samples are taken, with each result counted as a vote to move the sample point either earlier or later. These eight data bits constitute the voting window. The eight votes are then counted, and an action to adjust the position of the sampling clock occurs if there is a majority of early or late votes. The first-order algorithm makes a single phase adjustment per majority vote. The second-order algorithm acts repeatedly according to the net difference between early and late majority votes, thereby adjusting for the rate of change of phase.

Setting the ALIGN field to 01 enables alignment to the K28 comma symbols included in the 8b:10b data encoding scheme defined by the IEEE and employed by numerous transmission standards. For systems which cannot use comma-based symbol alignment, the single-bit alignment jog capability provides a means to control the symbol realignment features of the receiver directly from logic implemented in the ASIC core. This logic can be designed to support whatever alignment detection protocol is required.

The EQ bits allow for enabling and configuring the adaptive equalizer incorporated in all of the receive channels, which can compensate for channel insertion loss by attenuating the low-frequency components with respect to the high frequency components of the signal, thereby reducing inter-symbol interference. Above the zero frequency, the gain increases at 6 dB per octave until it reaches the high frequency gain. When enabled, the receiver equalization logic analyzes data patterns and transition times to determine whether the low-frequency gain of the equalizer should be increased or decreased. For the fully adaptive

setting (EQ = 0001), if the low-frequency gain reaches the minimum value, the zero frequency is then reduced. Likewise, if it reaches the maximum value, the zero frequency is then increased. This decision logic is implemented as a voting algorithm with a relatively long analysis interval. The slow time constant that results reduces the probability of incorrect decisions, but allows the equalizer to compensate for the relatively stable response of the channel.

- No adaptive equalization. The equalizer provides a flat response at the maximum gain. This setting may be appropriate if jitter at the receiver occurs predominantly as a result of crosstalk rather than frequency-dependent loss.
- Fully adaptive equalization. Both the low-frequency gain and zero position of the equalizer are determined algorithmically by analyzing the data patterns and transition positions in the received data.
- This setting should be used for most applications.
- Partially adaptive equalization. The low-frequency gain of the equalizer is determined algorithmically by analyzing the data patterns and transition positions in the received data. The zero position is fixed in one of eight zero positions. For any given application, the optimal setting is a function of the loss characteristics of the channel and the spectral density of the signal, as well as the data rate, which means it is not possible to identify the best setting by data rate alone; although generally speaking, the lower the line rate, the lower the zero frequency required.

2.3.1.3 Enabling the Transmitter

To enable a transmitter for serialization, the ENTX bit of the associated SERDES_CFGTX *n*_CNTL registers must be set high. When ENTX is low, all digital circuitry within the transmitter is disabled and clocks gated off, with the exception of the transmit clock (TXBCLK[*n*]) output, which continues to operate normally. All current sources within the transmitter will be fully powered down, with the exception of the current-mode logic (CML) driver, which remains powered up if boundary scan is selected. The Figure 3-4 shows the fields of SERDES_CFGTXn_CNTL.

2.3.1.4 SerDes Configuration Example

Example 2-1. SerDes Configuration

```
/* Set RX/TX config values based on the lane rate specified */

switch (linkRateGbps)

{

 case srio_lane_rate_5p000Gbps: /* Pll setting determines 5.0 Gpbs or 3.125
           Gbps */

 case srio_lane_rate_3p125Gbps: /* Same Tx and Rx settings for 5.0 Gbps or 3.125
             Gbps */

  rxConfig = 0x00440495;

 // (0) Enable Receiver

 // (1-3) Bus Width 010b (20 bit)

 // (4-5) Half rate. Two data samples per PLL output clock cycle

 // (6) Normal polarity

 // (7-9) Termination programmed to be 001

 // (10-11) Comma Alignment enabled
```

Example 2-1. SerDes Configuration (continued)

```
// (12-14) Loss of signal detection disabled

// (15-17) First order. Phase offset tracking up to +-488 ppm

// (18-20) Fully adaptive equalization

// (22) Offset compensation enabled

// (23-24) Loopback disabled

// (25-27) Test pattern mode disabled

// (28-31) Reserved

 txConfig = 0x00180795;

// (0) Enable Transmitter

// (1-3) Bus Width 010b (20 bit)

// (4-5) Half rate. Two data samples per PLL output clock cycle

// (6) Normal polarity

// (7-10) Swing max.

// (11-13) Precursor Tap weight 0%

// (14-18) Adjacent post cursor Tap weight 0%

// (19) Transmitter pre and post cursor FIR filter update

// (20) Synchronization master

// (21-22) Loopback disabled

// (23-25) Test pattern mode disabled

// (26-31) Reserved

 break;

case srio_lane_rate_2p500Gbps: /* Tx and Rx settings for 2.50 Gbps */

 rxConfig = 0x004404A5;

// (4-5) Quarter rate. One data sample per PLL output clock cycle

 txConfig = 0x001807A5;

// (4-5) Quarter rate. One data sample per PLL output clock cycle

 break;
```

Example 2-1. SerDes Configuration (continued)

```
 case srio_lane_rate_1p250Gbps: /* Tx and Rx settings for 1.25 Gbps */

  rxConfig = 0x004404B5;

 // (4-5) Eighth rate. One data sample every two PLL output clock cycles

  txConfig = 0x001807B5;

 // (4-5) Eighth rate. One data sample every two PLL output clock cycles

  break;

 default: /* Invalid SRIO lane rate specified */

  return -1;

}
```

The four SerDes lanes available on the TI SerDes module are configurable for various port widths, as shown in Table 2-5. When configured to 1x mode, a port consumes 1 transmit and 1 receive lane, and runs at the normal line rate specified through the serdes cfgpll, cfgtx, and cfgrx registers. Configuring a port for 2x mode consumes twice the amount of lanes and runs at twice the speed, while a 4x configuration ties all available lanes to a single port and runs at 4 times the line rate.

Table 2-5. SerDes Lane Configurations for Various Port Widths

	Lane A	Lane B	Lane C	Lane D	
Mode 0	1x				**Configuration 1**
Mode 0	1x	1x			**Configuration 2**
Mode 1	2x				
Mode 0	1x	1x	1x	1x	**Configuration 4**
Mode 1	2x		1x	1x	
Mode 2	1x	1x	2x		
Mode 3	2x		2x		
Mode 4	4x				

The port widths are controlled through the RapidIO PLM Port {0..3} Path Control registers.

NOTE: SerDes module information for KeyStone II devices is not provided in this user guide. Check for availability of the *SerDes User Guide for KeyStone II Devices* (SPRUHO3) on the device product page.

2.3.2 Direct I/O Operation

The direct I/O (load/store) module serves as the source of all outgoing direct I/O packets. With direct I/O, the RapidIO packet contains the specific address where the data should be stored or read in the destination device. Direct I/O requires that a RapidIO source device keep a local table of addresses for memory within the destination device. Once these tables are established, the RapidIO source controller uses this data to compute the destination address and insert it into the packet header. The RapidIO destination peripheral extracts the destination address from the received packet header and transfers the payload to memory through the DMA.

When a CPU is to send data from memory to an external processing element (PE), or read data from an external PE, it provides the RIO peripheral vital information about the transfer, such as DSP memory address, target device ID, target destination address, packet priority, and so forth. Essentially, a means must exist to fill all the header fields of the RapidIO packet. The load/store module provides a mechanism to handle this information exchange through a set of MMRs acting as transfer descriptors. These registers, shown in Figure 2-6, are addressable by the CPU through the configuration bus. There are 8 LSU in total. Each LSU has its own set of seven registers. LSU_Reg0-4 is used to store Control information, LSU_reg5-6 for Command and Status information. All these registers are RW except for LSU_REG6, which has a RO and a WO view. Upon completion of a write to LSUn_REG5, a data transfer is initiated for either an NREAD, NWRITE, NWRITE_R, SWRITE, ATOMIC, or MAINTENANCE RapidIO transaction. Some fields, such as the RapidIO srcTID/targetTID field, are assigned by hardware and do not have a corresponding command register field.

Figure 2-6. Load/Store Registers for RapidIO

	31-0							
LSU_Reg0	RapidIO Address MSB							
LSU_Reg1	RapidIO Address LSB/Config_Offset							
LSU_Reg2	DSP Address							
LSU_Reg3	31			30-20		19-0		
	Drbll_val			RSVD		Byte_Count		
LSU_Reg4	31-16	15-12	11-10	9-8	7-4	3-2	1	0
	DestID	SrcID_MAP	ID_Size	OutPortID	Priority	Xambs	Sup_gint	Int_Req
LSU_Reg5	31-16		15-8		7-4		3-0	
	Drbll_Info		Hop Count		FType		TType	
LSU_Reg6 (RO)	31	30	29-5		4		3-0	
	Busy	Full	RSVD		LCB		LTID	
LSU_Reg6 (WO)	31-28	27	26-6		5-2		1	0
	PrivID	CBUSY	RSVD		SrcID_MAP		Restart	Flush

The mapping of LSU register fields to RapidIO packet header fields is explained in Table 2-6, which has fields of the control and command registers (LSUn_REG0 to LSUn_REG5).

Table 2-6. Control/Command Register Field Description

Reg	Control/Command Register Field	Reset value	RapidIO Packet Header Field
LSU_REG0	RapidIO Address MSB	32'h0	32b *Ext Destination Address* Fields — Packet Types 2, 5, and 6
LSU_REG1	RapidIO Address LSB/Config_offset	32'h0	1) 32b *Destination Address* — Packet Types 2, 5, and 6 (used in conjunction with BYTE_COUNT to create 64b-aligned RapidIO packet header address) 2) 24b *Config_offset* Field — Maintenance Packets Type 8 (used in conjunction with BYTE_COUNT to create 64b-aligned RapidIO packet header Config_offset); The 2 lsb of this field must be zero because the smallest configuration access is 4B.
LSU_REG2	DSP Address	32'h0	32b DSP source address. NA for RapidIO Header
LSU_REG3	Byte_Count	19'h0	Number of data total bytes to read/write - up to 1MB (used in conjunction with RapidIO destination address to create WRSIZE/RDSIZE and WDPTR in RapidIO packet header). • 0x0000 — 1MB • 0x0001 — 1B • 0x0010 — 2B • … • 0xFFFFF — 1048575B Maintenance requests are limited to 4B or multiple word quantities
	Drbll_val	1'b0	For FType != 10 0h = There is no doorbell information which must be sent out at the end of the packet 1h = Doorbell information is valid If no response is required, send a doorbell after sending the last segment If response is required, after all responses are received with no errors, generate a doorbell with Drbll_Info If response is required, if response is received with error, doorbell is not generated.
LSU_REG4	Interrupt Req	1'b0	NA for RapidIO Header CPU-controlled request bit used for interrupt generation. Typically used in conjunction with non-posted commands to alert the CPU when the requested data/status is present. 0b0 - An interrupt is not requested upon completion of command 0b1- An interrupt is requested upon completion of command
	SUP_GINT	1'b0	Suppress good interrupt. If an interrupt request is set, and this bit is 0h = Interrupt generated on good completion 1h = Interrupt suppressed on a good completion
	Xambs	2'b0	RapidIO *xambs* field specifying extended address Msb
	Priority	4'b0	Priority = {VC, PRIO[1-0], CRF} **CRF** = RapidIO CRF field used in conjunction with the Priority bit. Packets have lower priority than a packet with same PRIO, VC bits but CRF = 1 Packets have higher priority than a packet with same PRIO, VC bits but CRF = 0 **PRIO** = 0-3 RapidIO prio field specifying packet priority. Request packets should not be sent at a priority level of 3, to avoid system deadlock. The software must assign the appropriate out-going priority. • 00 — Priority of 0 • 01 — Priority of 1 • 10 — Priority of 2 • 11 — Priority of 3 (cannot be used by the LSU) **VC**- RapidIO virtual channel bit, VCs not supported currently 1'b0- (TI only supports VC0 currently)

Table 2-6. Control/Command Register Field Description (continued)

Reg	Control/Command Register Field	Reset value	RapidIO Packet Header Field
	OutPortID	2'b0	NA for RapidIO Header. Indicates the output port number for the packet to be transmitted from. Specified by the CPU along with NodeID.
	ID Size	2'b0	RapidIO *tt* field specifying 8 or 16bit deviceIDs • 0b00 — 8b deviceIDs • 0b01 — 16b deviceIDs • 0b10 — reserved • 0b11 — reserved
	SrcID_MAP	4'h0	Defines which sourceID register** to use for this transaction • 0b0000 — Uses contents of RIO_DEVICEID_REG0 register • 0b0001 — Uses contents of RIO_DEVICEID_REG1 register • 0b0010 — Uses contents of RIO_DEVICEID_REG2 register • 0b0011 — Uses contents of RIO_DEVICEID_REG3 register • 0b0100 — Uses contents of RIO_DEVICEID_REG4 register • 0b0101 — Uses contents of RIO_DEVICEID_REG5 register • 0b0110 — Uses contents of RIO_DEVICEID_REG6 register • 0b0111 — Uses contents of RIO_DEVICEID_REG7 register • 0b1000 — Uses contents of RIO_DEVICEID_REG8 register • 0b1001 — Uses contents of RIO_DEVICEID_REG9 register • 0b1010 — Uses contents of RIO_DEVICEID_REG10 register • 0b1011 — Uses contents of RIO_DEVICEID_REG11 register • 0b1100 — Uses contents of RIO_DEVICEID_REG12 register • 0b1101 — Uses contents of RIO_DEVICEID_REG13 register • 0b1110 — Uses contents of RIO_DEVICEID_REG14 register • 0b1111 — Uses contents of RIO_DEVICEID_REG15 register ** Note that RIO_DeviceID_Regn are an input to the logic layer, and are not its memory map.
	DestID	16'b0	RapidIO *destinationID* field specifying target device
LSU_REG5	TType	4'b0	Transaction Field, relevant only for Ftype 2, 5, 8
	FType	4'b0	*ftype* field for all packets • 2 -> NREAD, Atomic instruction (TType for more details) • 5 -> NWRITE, NWRITE_R, Atomic (TType for more details) • 6 -> SWRITE • 8 -> Maintenance • 10 -> Doorbell All other encodings are reserved. If one of the reserved encoding is used, a CC of 0b100 is sent to indicate an error occurred.
	Hop Count	8'hFF	RapidIO hop_count field specified for Type 8 maintenance packets
	Drbll Info	16'h0	RapidIO doorbell info field for type 10 packets

Table 2-7 has fields of the status register (LSUn_REG6).

Table 2-7. Status Register Field Descriptions Read-Only View

Bit	Status Field	Reset Value	Function
31	BUSY	1'b0	Indicates status of the command registers 0b0 - Command registers are available (writable) for next set of transfer descriptors 0b1 - Command registers are busy with current transfer
30	Full	1'b0	Indicates that all shadow registers are in use 0b0 — At least 1 shadow register is available to be written 0b1 — No shadow registers are available to set up any transaction
29-5	Reserved	All 0's	Reserved
4	LCB	1'b1	LSU Context Bit. This information is used by the transaction to identify if the context of the CC is with respect to the current transaction or not.
3:0	LTID	4'b0	LSU Transaction Index. An LSU can support more than 1 transaction. This index helps identify the completion code (CC) information for the transaction.

Table 2-8. Status Register Field Descriptions Write-Only View

Bit	Status Field	Reset Value	Function
31-28	PrivID	4'b0	When attempting to manually release the busy bit, the PrivID of the original locking CPU must match the PrivID being used for release.
27	CBusy	1'b0	1h = Clear the busy bit if the PrivID matches 0h = No action
26-6	Reserved	All 0's	Reserved
5-2	SrcID_MAP	4'b1	Written by the software to indicate the SRCID of shadow register to be flushed.
1	Restart	1'b0	If an LSU is frozen on an error condition, a write of 1 to this bit restarts the LSU from the transaction where it was working on before the error condition occurred.
0	Flush	1'b0	If an LSU is frozen on an error condition, a write of 1 to this bit flushes all the shadow registers that match the SRCID in bits. This bit takes higher priority over the Restart bit.

Behind each LSU is a set of shadow registers. Each shadow register can be programmed for setting up a transaction ahead of time. When the LSU is free, the data is transferred from the shadow register to the LSU register. The point at which the data is actually transferred from the shadow registers to the actual LSU registers, the values of LSU0-5 registers can be read. Prior to that, a read of these registers indicates the last value present in these registers. This implies that the new values cannot be read until a write to REG5, and are further constrained by the fact that there is no other command ahead of the current command being setup. Reg6 can be read as and when it is written to, as it is used for locking and releasing.

The total shadow registers that can be assigned to each LSU is configurable, for a total of 32 sets of registers. Of the 32 total shadow registers, 16 are used for LSU0-3, while the other 16 can only be used for LSU4-7. The total shadow registers assigned to an LSU is set up through the RIO_LSU_SETUP_REG0.

NOTE: *Section 2.3.2.1* must be followed when writing to LSU_SETUP_REG0.

2.3.2.1 Writing to LSU_SETUP_REG0

Step 1. Write 1 to the BLK0_EN register (ensure GBL_EN is set to 1 before this step).

Step 2. Disable the LSU block by writing 0 to the BLK1_EN register.

Step 3. Poll for the BLK1_EN_STAT register to be 0 to make sure that the LSU is disabled.

Step 4. Write to the LSU_SETUP_REG0.

Step 5. Enable the LSU by writing 1 to the BLK1_EN registers.

The following restrictions are also applied on the shadow registers:

- Each LSU must have at least one shadow register at the minimum.
- No more than 9 shadow registers can be allocated to an LSU. A cnt of 001 => 1 shadow register is in use, 1001 => 9 shadow register. Sum total of all registers for LSU0-3 cannot be more than 16. The software must ensure this.
- The shadow registers do not have a unique MMR mapping. They take the address of the LSU they are assigned to.

Table 2-9 lists the pre-defined combinations for the shadow registers. No other combination is supported. There is a setup register associated with the 32 shadow registers. The configuration number is written to the setup register. Based on the configuration number, the hardware allocated the shadow registers for each LSU. The configuration number can be different for LSU0-3 and LSU4-7. This register is only programmable while the LSU is disabled and while the peripheral is enabled.

Table 2-9. Shadow Register Combinations

Configuration #	LSU0/LSU4	LSU1/LSU5	LSU2/LSU6	LSU3/LSU7
5'h00	4	4	4	4
5'h01	5	5	5	1
5'h02	5	5	4	2
5'h03	5	5	3	3
5'h04	5	4	4	3
5'h05	6	6	3	1
5'h06	6	6	2	2
5'h07	6	5	4	1
5'h08	6	5	3	2
5'h09	6	4	4	2
5'h0A	6	4	3	3
5'h0B	7	6	2	1
5'h0C	7	5	3	1
5'h0D	7	5	2	2
5'h0E	7	4	4	1
5'h0F	7	4	3	2
5'h 10	7	3	3	3
5'h 11	8	6	1	1
5'h 12	8	5	2	1
5'h 13	8	4	3	1
5'h 14	8	4	2	2
5'h 15	8	3	3	2
5'h 16	9	5	1	1
5'h 17	9	4	2	1
5'h 18	9	3	3	1
5'h 19	9	3	2	2

Figure 2-7. RIO_LSU_SETUP_REG0 (WO View)

31 — 21	20 — 16	15 — 5	4 — 0
Reserved	Shadow_grp1	Reserved	Shadow_grp0
W	W	W	W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-10. RIO_LSU_SETUP_REG0 (WO View) Field Descriptions

Field	Access Type	Reset Value	Description
Shadow_grp0	WO	5'h0	The total number of shadow registers associated with all the LSU 0-3 based on Table 2-9.
Shadow_grp1	WO	5'h0	The total number of shadow registers associated with all the LSU 4-7 based on Table 2-9.
Reserved	—	0s	These bits are reserved. Any write to the reserved bits is ignored.

Figure 2-8. RIO_LSU_SETUP_REG0 (RO View)

31 — 28	27 — 24	23 — 20	19 — 16	15 — 12	11 — 8	7 — 4	3 — 0
Lsu7_cnt	Lsu6_cnt	Lsu5_cnt	Lsu4_cnt	Lsu3_cnt	Lsu2_cnt	Lsu1_cnt	Lsu0_cnt
R	R	R	R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-11. RIO_LSU_SETUP_REG0 (RO View) Field Descriptions

Field	Access Type	Reset Value	Description
LSUx_cnt	RO	4'h4	The total number of shadow registers associated with the LSUx. This register is only programmable while the LSU logic is disabled. The read-only view reflects the total shadow registers that are associated with each. Legal range is 4'h1-4'h9. All other values are reserved.

The recommended models for using the LSU are as follows:

- LSU setup by cores:
  - Each core can have fixed number of shadow registers allocated to the LSU. For example, if an LSU has 6 shadow registers, the software must dedicate 4 of these to core0 and the other 2 to core1. This ensures that the registers are shared fairly between the cores, and that one of the cores never has starvation for the shadow registers. When the shadow register of a core is released, that core can setup a new transaction in the shadow register.
  - All shadow registers of an LSU are dedicated to a single core.
  - TI recommends that the user allocate all transactions to a specific LSU at the same priority. This helps alleviate HOL blocking issues in the LSU.
- LSU setup by EDMA

  This model can only support one EDMA channel on a specific LSU dedicated to a single core. The LSU cannot be shared. Only one shadow register may be assigned to this LSU. However, there is no restriction if the user wishes to use more than one shadow register.

Each shadow register set has a copy of LSU_Reg0-5. LSU_Reg6 is shared between all of them for a specific LSU. An LSU transaction is setup up by writing to LSU_Reg0-5. This write actually occurs in the shadow register. If the LSU HW is free, the next available shadow registers are picked up and a data transfer is initiated for an NREAD, NWRITE, NWRITE_R, SWRITE, ATOMIC, or MAINTENANCE RapidIO transaction. If the LSU HW is already working on a current shadow register set, the pending LSU transactions within the remaining shadow registers should wait until the current transaction is completed. As soon as the LSU HW finishes with the current transaction, it is loaded with the next available set of shadow register information. Therefore, if an LSU has 3 shadow registers assigned to it, it can have one active and 2 pending transactions in its queues.

2.3.2.2 Full Bit

Before a CPU can try to write to a shadow register it must check for availability of shadow registers. The CPU must do a read of LSU_REG6 to check the status of full bit. If the bit is a 1, there are no more shadow registers available. If the bit is a 0, there is at least one shadow register available.

The full bit is set only when a write to LSU_REG5 is completed for the last free shadow register. Thus the full and busy bits cannot be set at the same time.

2.3.2.3 Busy Bit

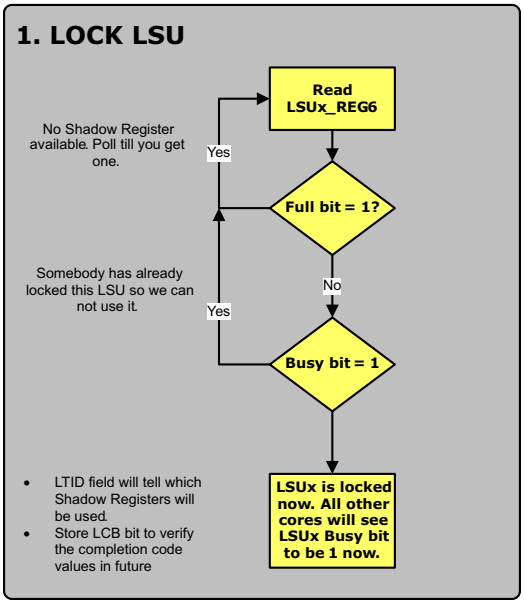
To ensure that two CPUs are not trying to access the same set of registers at the same time, a busy bit is present in LSU_REG6. When the CPU is trying to grab an LSU it follows these three steps:

1. Lock the registers: It reads LSU_REG6 to ensure that the busy and the full bits are 0. The following can happen:
   a. Full bit is set: This implies no shadow register is available. The CPU must read the register again to check the availability of a shadow register. This case should not happen if the software restricts the total shadow register than can be used by each core.
   b. Some other CPU has the LSU locked: In a multicore environment, it is possible that the LSU is locked by another core. In this case, the read of LSU_Reg6 indicates that the busy bit is 1. The core must try and read the LSU_Reg6 again in an attempt to lock it. An EDMA does not require this read, as there is no possibility of ownership conflict because it has a dedicated LSU. It can be determined whether an LSU is dedicated to an EDMA; see Figure 2-16.
   c. Normal: The read shows the busy bit = 0, which implies that the core now has the LSU registers locked for its use. The hardware saves off the PRIVID of the master that locked the shadow register. The busy bit is set to a 1. The only exception to this is if the read was from the debugger side. In that case, though the value of the register will be returned, no other action results from the read of this register.

      If a particular master locks an LSU and the same master reads the LSU BUSY bit in some other context (for example, in an ISR), then the busy bit always reads as 1 and not 0. Thus, the LSU busy bit can be used to share a particular LSU shadow register set among various tasks of the same master. Once the master completes writing to LSU_REG5, the busy bit is cleared.
      The read of LSU_Reg6 returns the following:
      i. LTID (LSU transaction ID). This is a temporary ID attached to the transaction. At the end of the transaction, the software can use this ID to read the relevant completion code. This ID cannot exceed the maximum number of shadow registers mapped to the LSU. Thus for LSU0, If the maximum shadow registers allocated were 4, then LTID can only range from 0 to 3.
      ii. LCB (LSU context bit): This gives frame of reference for status bits, whether it is for the current transaction, the previous one, or the next one. After reset, when an LSU grabs the lock for a specific LTID, the LCB returned is a 1. The next time an LSU grabs the lock for the same LTID, the LCB returned is a 0. Therefore, if this value is a 1 on a read, and the completion code indicates an LCB of 0, the software knows that the completion information is not that the transaction of interest. After reset, the first LCB for an LTID provided will be a 1.

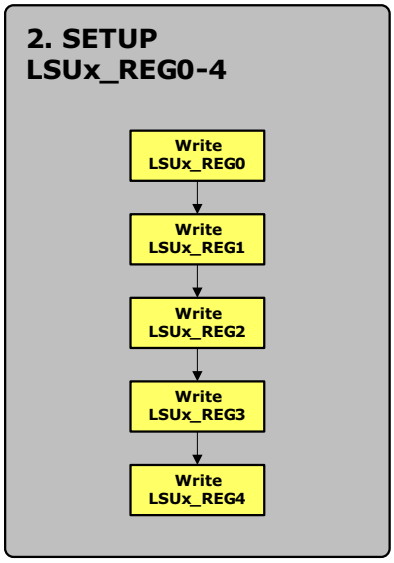
NOTE: The BUSY and FULL bits must be read simultaneously from LSU_REG6. The reason for this is that as soon as soon as the master checks for the FULL bit status and obtains permission to use the LSU registers, the busy bit is subsequently set to 1. Trying to poll the busy bit afterwards always results in the bit being set.

Figure 2-9. Lock LSU Registers



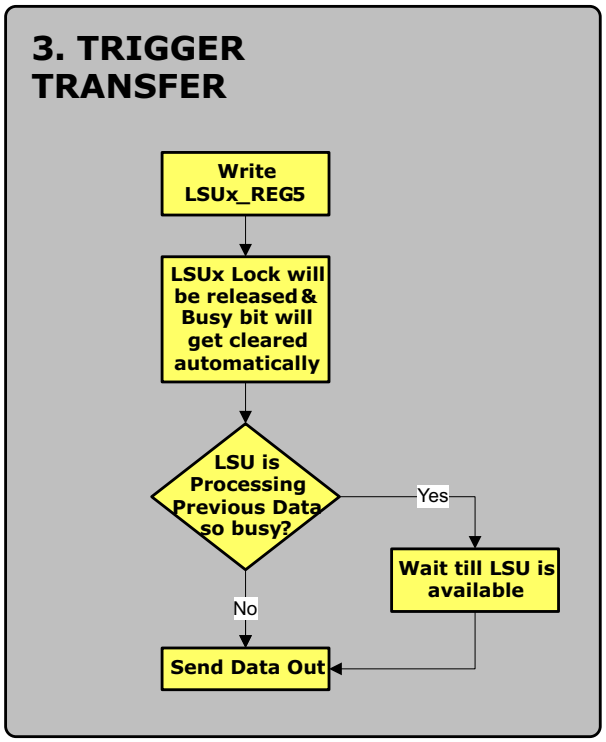
2. Setup the registers: Reg0–Reg4 are setup by the CPU. The PRIVID of these write requests must match the PRIVID of the original request that locked the LSU busy bit, else the write request is ignored. If the LSU is being used by an EDMA, the PRIVID check is ignored. While the registers are being setup, the LSU could be busy handling another request at this time.

Figure 2-10. Set Up LSU Registers



3. Release the lock: A final write to Reg5 is done to clear the busy bit. The shadow register is now ready to be used by the LSU. If the LSU is already busy, the data is simply held in the shadow register. When the LSU finishes with the transaction, the shadow register is ready to be used by the LSU.

Figure 2-11. Release LSU And Trigger Transfer



4. LSU locked due to a hang: If the CPU that locked the LSU does not come back to finish the writes of the rest of the registers, and the final write to LSU5, the LSU can get locked. A CPU trying to lock the shadow register of an LSU could keep polling the REG6. If, after several attempts, it finds that the LSU is always locked as the busy bit is set (which implies that the Full bit cannot be set), and the LTID and the LCB are staying the same, the CPU can infer that the LSU is locked by a device that is no longer accessing the LSU. To overcome this issue, a CBUSY (clear busy) bit is present in the WO version of LSU_REG. A master on the device can intervene and set the CBUSY bit, along with the PRIVID information. If the PRIVID matched the information that came when the LSU was locked originally, the hardware releases the busy bit and sets the CC bits to 111. This specific shadow register writes are now complete. The next lock of the busy bit will get the next LTID and LCB, and not reuse the current LTID and LCB.

The maximum LTID that can be supported per LSU are as follows:

Table 2-12. Maximum LTID per LSU

LSU#	LTID
0/4	9 (0-8)
1/5	6 (0-5)
2/6	5 (0-4)
3/7	4 (0-3)

Additional registers RIO_LSU_STAT_REG0-2 are required to keep track of the completion codes (CC) of the various transactions. The LTID is used to tie the information back to the CC bits. The software knows which LSU the transaction was assigned to, so it knows which bits of the RIO_LSU_STAT_REG0-2 to look at. The LCB in the register informs the software if the reference is for the current transaction, the previous one, or the next one. This information is required if the software is polling the CC bits. Thus, the status bits for a shadow register are as follows. A similar set of three registers (RIO_LSU_STAT_REG3-5) are also present for LSU4-7. When the CPU issues a flush or restart command (described in Table 2-13) for an LSU, the context-specific Completion Code bits for that LSU are reset back to 000. The reset values of all RIO_LSU_STAT_REG0-5 registers are zeros.

Table 2-13. LSU_STATUS (LSUx_STAT)

Control/Command Register Field	Access Type	Reset value	Description
Completion Code	RO	3'b0	Indicates the status of the pending command 0b000 — Transaction complete, No Errors (Posted/Non-posted) 0b001 — Transaction Timeout occurred on Non-posted transaction 0b010 — Transaction complete, Packet not sent due to flow control blockade (Xoff) 0b011 — Transaction complete, Non-posted response packet (type 8 and 13) contained ERROR status, or response payload length was in error 0b100 — Transaction complete, Packet not sent due to unsupported transaction type or invalid programming encoding for one or more LSU register fields 0b101 — DMA data transfer error 0b110 — "Retry" DOORBELL response received, or Atomic Test-and-swap was not allowed (semaphore in use) 0b111 — Transaction completed, No packets sent as the transaction was killed using the CBUSY bit. If the doorbell is also requested in the command, the completion code is set either as soon as an error condition is encountered or after the response of the doorbell transaction is received.
LSU Context Bit	RO	1'b0	Gives the reference of the current CC bits. For a transaction, this need the match the LCB that was returned on a read of LSU_Reg6

Table 2-14. RIO_LSU_STAT_REG0-6 & RIO_LSU_STAT_REG3-5

31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0
Lsu0_Stat7	Lsu0_Stat6	Lsu0_Stat5	Lsu0_Stat4	Lsu0_Stat3	Lsu0_Stat2	Lsu0_Stat1	Lsu0_Stat0
Lsu2_Stat0	Lsu1_Stat5	Lsu1_Stat4	Lsu1_Stat3	Lsu1_Stat2	Lsu1_Stat1	Lsu1_Stat0	Lsu0_Stat8
Lsu3_Stat3	Lsu3_Stat2	Lsu3_Stat1	Lsu3_Stat0	Lsu2_Stat3	Lsu2_Stat2	Lsu2_Stat1	Lsu2_Stat0

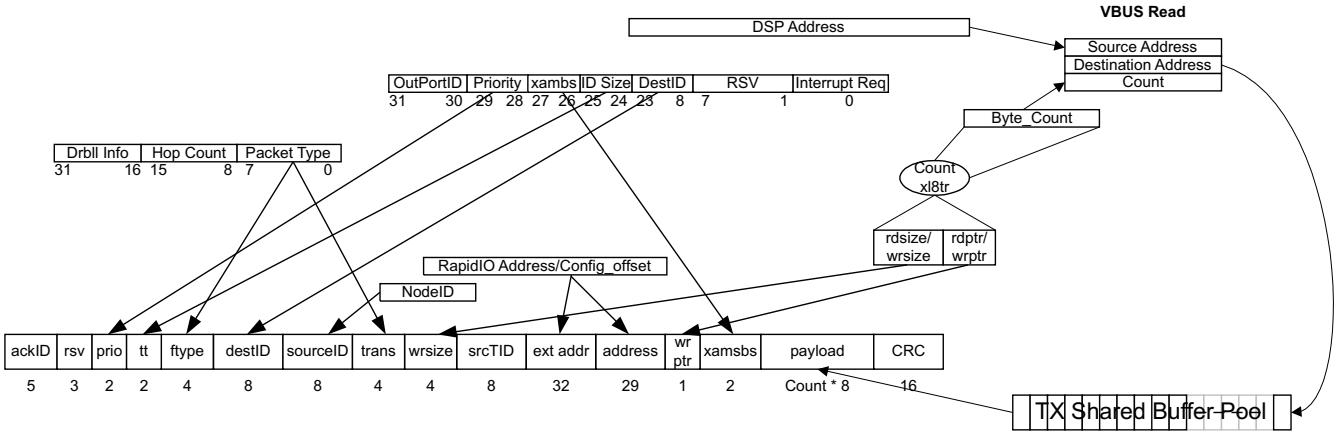
All the interrupts associated with the errors above are reported immediately. In case of error code 011, the error is reported right away. No new transaction is submitted. However, the LSU is freed up only after:

- CPU writes the restart or flush bit.
- All responses for the outstanding non-posted transactions have been received, or a response timeout occurred in getting the response. If there is a response timeout, no additional interrupt or CC bits are set for this response timeout. If all responses are received and the restart command is received, the LSU flushes only the current transaction and loads the next set of shadow registers. If the user writes a flush and all responses are received, the user then flushes out all the transactions of the specific SRCID_MAP in the shadow register of the specific LSU. If neither a restart nor flush command is received, and all the responses are received, the peripheral waits until it gets a CPU restart/flush timeout. The LSU discards all transactions in the shadow registers, as well as the current transaction that caused the error that originated from the same SRCID_MAP. It then reloads the next shadow register on its own. For more information, see *Section 2.3.2.10*.

There are eight LSU register sets. This allows eight outstanding requests for all transaction types that require a response (that is, non-posted). For multicore devices, software manages the usage of the registers. A shared configuration bus (VBUSP interface) is used to access all register sets. A single core device can utilize all eight LSU blocks.

Figure 2-12 shows an example of the data flow and field mappings for a burst NWRITE_R transaction:

Figure 2-12. Example Burst NWRITE_R



For WRITE commands, the payload is combined with the header information from the control and command registers, and buffered in the shared TX buffer resource pool. Finally, it is forwarded to the TX FIFO for transmission. READ commands have no payload. In this case, only the control and command register fields are buffered and used to create a RapidIO NREAD packet, which is forwarded to the TX FIFO. Corresponding response packet payloads from READ transactions are buffered in the shared RX buffer resource pool when forwarded from the receive ports. Both posted and non-posted operations rely on the OutPortID command register field to specify the appropriate output port/FIFO.

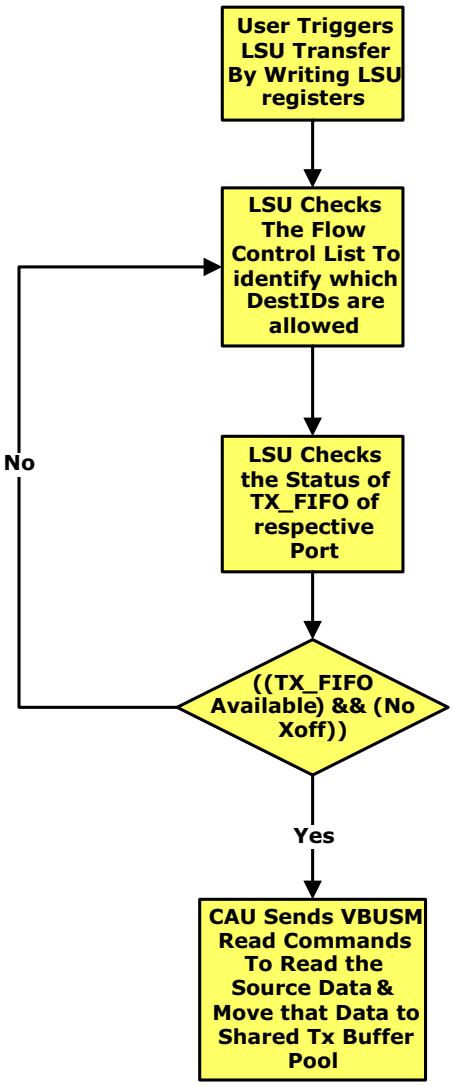
The data is burst internally to the load/store module at the DMA clock rate.

2.3.2.4 Detailed Data Path Description

The load/store module is used for generating outgoing RapidIO Direct I/O packets. This interface does not support messaging interface. In addition, outgoing DOORBELL packets are generated through this interface. Each LSU can support up to 16 SRCTIDs. Thus LSU0 can generate transactions with SRCTID 0-15, LSU1 from SRCTIDs 16-31, and so on. When a new command is started from an LSU, it starts back at the starting SRCTID. Thus for LSU1, every new command starts at 16.

The data path for this module uses VBUSM as the DMA interface. The SRIO maximum payload size is 256B. Each LSU may possibly generate more than one VBUSM transaction to get payloads bigger than 256B in parallel. These payloads can then be sent on the UDI interface. However, to distinguish between these transactions, it is important to have different SRCTIDs. Even though the data is going to the same LSU, the transactions can then be distinguished from each other when the response is coming back from the UDI interface. Figure 2-13 shows the flow diagram of the operation of sending data out of the ports.

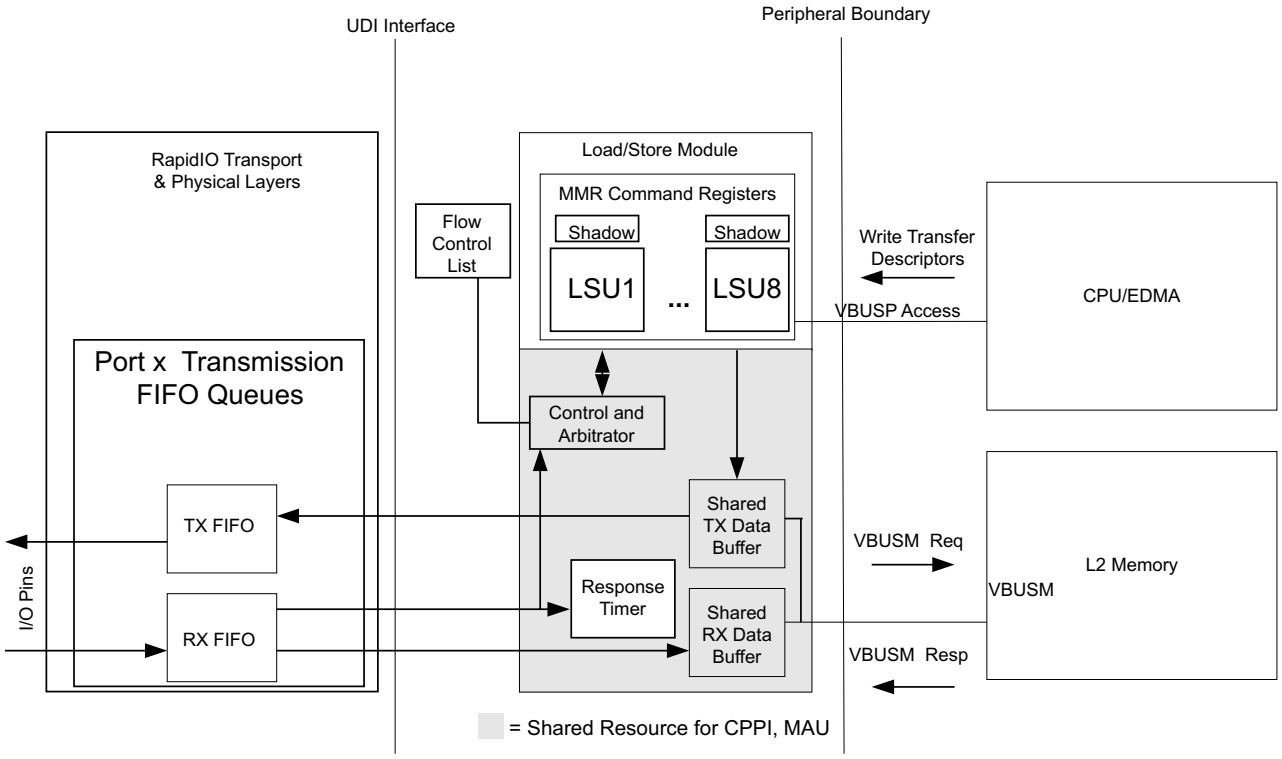
Figure 2-13. Packet Data Path



The VBUSP configuration bus is used by the CPU to access the control and command registers. The registers contain transfer descriptors required to initiate READ and WRITE packet generation. After the transfer descriptors are written, flow control status is queried. The unit examines the DESTID and PRIORITY fields of the command registers to determine if that flow has been Xoff'd. Additionally, the free buffer status of the TX FIFO is checked (based on the OutPortID command register field). Only after the flow control access is granted and a TX FIFO buffer has been allocated can a VBUSM read command be issued for payload data to be moved into the shared TX buffer pool. Data is moved from the shared buffer pool to the appropriate output TX FIFO in simple sequential order, based on completion of the VBUSM transaction. Once in the FIFO, the data is guaranteed to be transmitted through the pins.

Figure 2-14 shows the data path and buffering that is required to support the load/store module.

Figure 2-14. Load/Store Module Data Flow



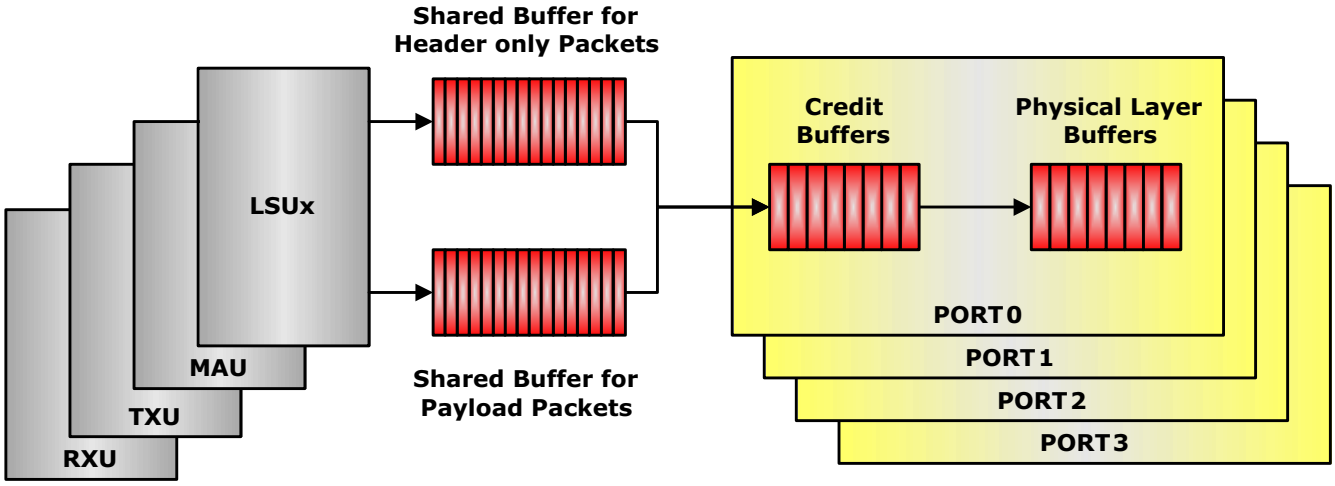
2.3.2.5 TX Operation

2.3.2.5.1 WRITE Transactions

Figure 2-15 shows the buffering mechanism inside the Serial RapidIO module. The shared TX data buffers are shared between multiple cores. A state machine arbitrates and assigns available buffers between the LSUs and other protocol units. The load/store module can only forward the packet to the TX FIFO after the final payload byte from the VBUSM response has been written into the shared TX data buffers. Once forwarded to the TX FIFO, the shared buffer can be released and made available for a new transaction.

The TX buffer space is dynamically shared among all outgoing sources, including the load/store unit (LSU), TX CPPI, and the response packets from RX CPPI and the memory access unit (MAU). Thus the buffer space memory must be partitioned to handle packets with and without payloads.

Figure 2-15. SRIO Buffering Mechanism



There can be 16 payload packets with payload size up to 256B, and 16 header packets without payload. The header packets are the packets without a payload (for example, response packets). Data leaves the shared buffer pool sequentially in order of receipt, and does not consider the packet priority. Priority can affect the order in which the data leaves the TX FIFOs.

For posted WRITE operations that do not require a RapidIO response packet, a core may submit multiple outstanding requests. For instance, a single core may have many streaming write packets buffered at any given time, given out-going resources. In this application, the LSU can be released to the shadow registers once the packet is written into the shared TX buffer pool. If the request has been flow controlled, the peripheral sets the completion code status register and appropriate interrupt bit of the ICSR. The control and command registers can be released after the interrupt service routine completes.

For non-posted WRITE operations that do require a RapidIO response packet, there can be only one outstanding request per core at any given time. The packet is written to the shared TX buffer pool as described above; however, the LSU can not be released until the response packet is routed back to the module and the appropriate completion code is set in the status register. One special case exists for Atomic out-going test-and-swap packets (Ftype 5, Transaction 0b1110). This is the only WRITE class packet that expects a response with payload. This response payload is routed to the LSU, it is examined to verify whether the semaphore was accepted, and then the appropriate completion code is set. The payload is not transferred out of peripheral through VBUSM.

Thus, the general flow would be:

- Command registers are written using the config bus (VBUSP)
- Flow control determined
- TX FIFO (TX shared buffer pool) free buffer availability determined
- VBUSM read request for data payload
- VBUSM response writes data to specified module buffer in the shared TX buffer space
- VBUSM read response is monitored for last byte of payload
- Header data in the command registers written to the shared TX buffer space
- Transfer payload and header to TX FIFO
- Load the next shadow register if no RapidIO response is needed
- Transfer from TX FIFO to external device based on priority

2.3.2.5.2 READ Transactions

The flow for generating READ transactions is similar to non-posted WRITE with response transactions. There are two main differences: 1) READ packets contain no data payload, and 2) READ responses have a payload. So READ commands simply require a non-payload TX buffer within the shared pool. In addition, they require a shared RX data buffer. This buffer is not pre-allocated before transmitting the READ request packet, as doing so could cause traffic congestion of other in-bound packets destined to other functional blocks.

As explained earlier, the LSU cannot be released until the response packet is routed back to the module and appropriate completion code is set in the status register.

Thus, the general flow would be as follows:

- Command registers are written using the config bus (VBUSP)
- Flow control determined
- Allocation of TX FIFO buffer
- Header data in the command registers written to the shared TX buffer
- Transfer header to TX FIFO
- Transfer from TX FIFO to external device based on priority
- When the data comes back on the UDI interface, the data is sent out through the VBUSM

For all transactions, the shared TX buffers are released as soon as the packet is forwarded to the TX FIFOs; if an ERROR or RETRY response is received for a non-posted transaction, the CPU must reinitiate the process by writing the LSU_Reg6, or initiate a new transaction altogether.

2.3.2.5.3 Packet Segmentation

The LSU handles two types of segmentation of outbound requests. First, when the Byte_Count of Read/Write requests exceeds 256Bytes. Second, when Read/Write request RapidIO address is non-64b aligned. In both cases, the outgoing request must be broken up into multiple RapidIO request packets. For example, the CPU wants to perform a 1KB store operation to an external RapidIO device: after setting up the LSU registers, the CPU simply performs one write to the LSU_Reg5 command register. The peripheral hardware then segments the store operation into four RapidIO write packets of 256B each, and calculates the 64b-aligned RapidIO address, WRSIZE, and WDPTR, as required for each packet. The LSU registers can not be released until all posted request packets are passed to the TX FIFOs. Alternatively, for non-posted operations, such as CPU loads, all packet responses must be received before the LSU registers are released.

2.3.2.6 RX Operation

Response packets are always type 13 RapidIO packets. All response packets with transaction types not equal to 0b0001, and SRCTID less than 128 are routed to the LSU block sequentially in order of reception. These packets may or may not have a payload depending on the type of corresponding request packet that was originally sent. Due to the nature of RapidIO switch fabric systems, response packets can arrive in any order. The data payload, if any, and header data is moved from the RX FIFO to the shared RX buffer. The *targetTID* field of the packet is examined to determine which core and corresponding set of registers are waiting for the response. Remember, there can be only one outstanding request per core. All payload data is moved from the shared RX buffer pool into memory through normal VBUSM operations.

The MAU block is responsible for all incoming direct I/O packets, including NREAD, SWRITE, NWRITE and NWRITE_R transactions. Incoming Atomic operations are not supported and are responded to with an ERROR response. In addition to direct I/O packets, the MAU is responsible for routing DOORBELL messages to the interrupt handler, as well as doing packet forwarding.

All incoming directIO packets contain a memory address field. The address is the location in the device's memory map from which the data will be written or read. There are no RX address translations windows supported by the peripheral. The exact address from the RapidIO packet will be used for the DMA transaction. This approach requires that the source of the directI/O transaction have detailed knowledge of destination device's memory map. Also, it should be noted that there is not a hardware memory protection

scheme for sources overwriting each other's data at a destination. This must be managed with software at a system level. Some memory accesses may be restricted at the device level to supervisor permissions. General access memory areas are accessible with user permissions. The MAU assigns user permissions to each received packet that is DMA'd, unless the SOURCEID of that packet matches the RIO_SUPRVSR_ID value. Supervisor permissions are assigned only if there is a match to this register.

2.3.2.7 Reset and Powerdown

Upon reset, the load/store module puts all the register fields in their default values and waits for a write by the CPU.

The load/store module can be powered down if the direct I/O protocol is not being supported in the application. For example, if the messaging protocol is being used for data transfers, it is beneficial to powerdown the load/store module to save power. In this situation, the command registers should be powered down and inaccessible. Clocks should be gated to these blocks while in the powerdown state.

2.3.2.8 Special Conditions

2.3.2.8.1 Timeout

Registers for all non-posted operations should only be held for a finite amount of time, to avoid blocking resources when a request or response packet is somehow lost in the switch fabric. This time correlates to the 24-bit port response time-out control CSR value discussed in Section 5.10.1 and 6.1.2.4 of the serial specification. If time expires, an error should be logged in the ERROR MANAGEMENT RapidIO registers. The response timeout is handled exactly like any other error condition. The LSU waits for the CPU to send it a flush or a restart command. The RapidIO specification states that the maximum time interval (all 1s) equates between three and six seconds. A logical layer timeout occurs if the response packet is not received before a countdown timer (initialized to this CSR value) reaches zero.

Each outstanding packet response timer requires a 4-bit register. The register is loaded with the current timecode when the transaction is sent. The timecode comes from a 4-bit counter associated with the 24 bit down counter that continually counts down and is reloaded with the value of SP_RT_CTL (address offset 1124h) when it reaches 0. Each time the timecode changes, a 4-bit compare is done to the register. If the register becomes equal to the timecode again, without a response being seen, then the transaction has timed out. Essentially, instead of the 24-bit value representing the period of the response timer, the period is now defined as P = (2^24 x 16)/F. This means the countdown timer frequency must be 44.7 to 89.5 Mhz for a 6 to 3 second response timeout. Because the needed timer frequency is derived from the DMA bus clock (which is device-dependent), the hardware supports a programmable configuration register field to properly scale the clock frequency. This configuration register field is described in the Peripheral Setting Control register.

2.3.2.8.2 Incoming Error Response

In the event that a response packet indicates ERROR status, the load/store module notifies the CPU by generating an error interrupt for the pending non-posted transaction. No new transaction will be sent out. The error interrupt/CC bits are sent out only after all responses have come back for the transactions that have already been submitted. In the event that the response completed successfully, and the Interrupt Req bit is set in the control register, the module generates a CPU servicing interrupt to notify the CPU that the response is available. The control and command registers can be released once the response packet is received by the logical layer.

2.3.2.8.3 Handling the SRCID Based LSU Error Interrupts in Software

The LTID and LCB information can only be read during the step, in which the LSU_REG6 is read with LSU_REG6.FULL = 0 and LSU_REG6.BUSY = 0, and the LSU resource is locked. Each of the masters (CorePACs) which can use a particular LSU are responsible to maintain the LTID and LCB information of all outstanding DIO transactions originating from it. The LTID is then used by the CorePAC to know exactly which field of the RIO_LSU_Status_Regs to look at to track the completion code and LCB. For example:

Consider LSU0, which is configured to have 4 shadow registers. If LSU0 is used only by CorePAC0, then CorePAC0 should maintain the history of the 8 latest transactions (for all possible combinations of LTIDs and LCB). When CorePAC0 wants to check the LSU0_Status, due to an interrupt for example, it then reads RIO_LSU_Status_Reg0, bits 15:0 (corresponding to shadow registers 3-0), and compares the history of outstanding transactions to the register field values. It can determine completion code based on LTIDn = Statn field and matching LCB within the register.

Now, if LSU0 is used by both CorePAC0 and CorePAC1, then both CorePAC0 and CorePAC1 should maintain a separate history of the 8 latest transactions (including all combinations of LTIDs and LCB) originating from it.

When the LSU shadow registers are used, the SRCID-based LSU interrupts (LSU_ICSR0) should be used. Here, each SRCID maps to a particular master (CorePAC) in the SoC. The good or bad interrupts for a particular SRCID are handled by the CorePAC associated with this SRCID. For better performance, suppress the good interrupts generated from the LSU, while configuring the LSU registers. Thus, mostly only the LSU bad or error interrupts must be handled.

When a CorePAC receives a SRCID-based error interrupt, it must read all the RIO_LSU_STAT_REGs associated with the LSU resources which it can configure for sending out DIO transactions. Once the LSU number, LTID, and LCB of the erroneous transaction are identified, this info is compared with the history to determine which LSU transaction resulted in error.

2.3.2.8.4 Incoming Retry Response for Doorbells

The hardware is not responsible for attempting retransmission of doorbell. Other non-posted transactions can only get a response of DONE or ERROR.

In the event that a doorbell response packet indicates Retry status, the load/store module notifies the CPU by generating an interrupt. The control and command registers can be released once the response packet is received by the logical layer. The hardware is not responsible for attempting retransmission of the doorbell transaction.

Thus, the general flow would be as follows:

- Previously, control and command registers were written and request packet sent
- Response packet Type13, Trans != 0b0001 arrives at module interface, and handled sequentially (not based on priority)
- Examine targetTID to determine routing of response to appropriate core
- Check status field of response packet for ERROR, RETRY, or DONE
- If DONE, submit a VBUSM request and transmit the payload (if any) to a DSP address. If ERROR or RETRY, set the interrupt.
- Optional interrupt to CPU notifying of packet reception

2.3.2.9 Scheduling

At any point of time multiple LSU can be ready for transactions. The following conditions are applied before a transaction is considered for scheduling:

- Priority, CRF: Only the highest priority and CRF transactions are considered for the scheduling for each port.
- Flow Control: The DESTID for the transaction should not be flow-controlled.
- SRCID for non-posted transactions: If the SRCID is already being used for a non-posted transaction and another non-posted transaction comes from the same SRCID, the request is suspended until the previous SRCID transaction is not complete.
- SRIO VBUSM limitation: The VBUSM interface can support up to 4 read and 4 write transactions, which are shared between the MAU, LSU, and TXU. If the LSU must send a command on the VBUSM interface, there should be availability in the VBUSM interface.
- FIFO: There should be space in the FIFO to be able to accommodate the transaction.

If multiple LSU meet the above qualifications, the scheduler switches between then at the maximum of SRIO packet size of 256B.

2.3.2.10 Error Handling

Various errors can occur that are reported in the CC bits of the RIO_LSU_STAT_REGx. Additionally, if the Int_req and the Sup_gcomp (impacts good completion interrupt bit only) bit is deasserted, the bit is set in LSU_Reg4, the interrupt is routed back to the CPU or EDMA. The information is generated for each SRCID specific in LSU_Reg3, as well as for all the eight LSU. On legacy devices, the information was only available per LSU. However, if multiple cores are using the LSU, the LSU interrupts all the cores, even though the completion has been for a single core only. Therefore, it is important to distinguish between the interrupts per core versus LSU. Because the EDMA is dedicated to an LSU, information-per-LSU is also needed. The RIO_LSU_SETUP_REG1 register is used to setup the information, whether the LSU is being used by an EDMA or not. This register is only programmable while the LSU is disabled.

Figure 2-16. RIO_LSU_SETUP_REG1

31 10	9 8	7	6	5	4	3	2	1	0
Reserved	Timeout _cnt	Lsu7 _edma	Lsu6 _edma	Lsu5 _edma	Lsu4 _edma	Lsu3 _edma	Lsu2 _edma	Lsu1 _edma	Lsu0 _edma
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 2-15. RIO_LSU_SETUP_REG1 Field Descriptions

Field	RW	Reset	Description
LSUx_edma	RW	1'b0	0h = Good completion will be driven to the interrupt based on the SRCID 1h = Good condition interrupt will be driven to the LSU specific interrupt bit
Timeout_cnt	RW	2'h0	00h = After an error condition, Timecode changes only once before the LSU transaction is discarded and a new transaction is loaded from the shadow register (refer to the RXU section for timecode infomation) 01h = After an error condition, Timecode changes 2 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register 10h = After an error condition, Timecode changes 3 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register 11h = After an error condition, Timecode changes 4 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register

For each transaction the following interrupts are generated. The software determines which of these interrupts to use and routes them accordingly:

- Good completion per SRCID or LSU based on the setup bits in Table 2-15.
- Error completion per SRCID defined in LSU_Reg4

In case of an error, the LSU is not automatically loaded with the next shadow registers. It waits for the software to intervene. When the CPU gets an interrupt, it reads the relevant RIO_LSU_STAT_REGx to determine the details of the error. Now the software can do one of the following:

- The software can choose to not do anything. In this case, a timer expires (based on the Timeout_cnt above). As the CPU does nothing and waits for the Timeout_cnt to expire, the hardware discards the current transaction that caused the issue. It then reloads the next shadow register on its own. If the LSU was setup for an EDMA, it sends a good completion interrupt to the EDMA, thus enabling the EDMA to automatically load the next transaction.
- The software fixes the issue (for example, enables the port that is XOffed), and sets a restart bit for the specific LSU. The LSU terminates the current transaction and loads the next set of shadow registers.
- The software decides to flush the transaction. It writes to the flush bit for the LSU. All transactions originating from the same SRCID present in the shadow registers for the specific LSU are flushed. It may take more than one cycle to perform the flush.

The flush bit and the restart bits are the write version of the LSU_Reg6.

2.3.2.11 DirectIO Programming Considerations

The steps in Figure 2-17 show the programming details of LSU. Here, it uses the CSL register layer APIs to perform register read and write.

Figure 2-17. Programming Details of the LSU

```
/*****************************************************************************************************************/
/***** STEP# 1: LOCK LSU ****************************************************************************************/
/*****************************************************************************************************************/

/***** Check the Full bit to check Whether all Shadow Registers are Full or Not if it is then keep on
polling till the time any Shadow Register becomes available ****/
full        = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_FULL);
while ((full & 0x00000001) == 0x00000001)
{
        full        = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_FULL);
}

/***** Check the Busy bit to check whether any other master has locked the LSU or not ****/
busy        = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_BSY);
while ((busy & 0x00000001) == 0x00000001)
{
        busy        = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_BSY);
}

/***** Store the LTID field to get the LSU Shadow Register Number ****/
LTID        = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_LTID);

/***** Store the LCB bit for Verifying the validity of the Completion Code ****/
LCB         = CSL_FEXT(SRIO_REGS->LSU[Lsuno].LSU_REG6, SRIO_LSU_REG6_LCB);

/*****************************************************************************************************************/
/***** THE CURRENT LSU IS LOCKED NOW ALL OTHER MASTERS WILL SEE BUSY BIT TO BE **********/
/*****************************************************************************************************************/

/*****************************************************************************************************************/
/***** STEP# 2: SETUP LSUx_REG 0 – 4 ****************************************************************************/
/*****************************************************************************************************************/

/***** Program Address For The Destination DSP Buffer ****/
SRIO_REGS->LSU[Lsuno].LSU_REG0 =   CSL_FMK( SRIO_LSU_REG0_ADDRESS_MSB,0 );
SRIO_REGS->LSU[Lsuno].LSU_REG1 =   CSL_FMK( SRIO_LSU_REG1_ADDRESS_LSB_CONFIG_OFFSET,(int )&rcvBuff[0] );

/***** Program Address For The Source DSP Buffer *****/
SRIO_REGS->LSU[Lsuno].LSU_REG2 =   CSL_FMK( SRIO_LSU_REG2_DSP_ADDRESS, (int )&xmtBuff[0]);

/***** Program The Payload Size& No Doorbell After Completion *****/
SRIO_REGS->LSU[Lsuno].LSU_REG3 =   CSL_FMK( SRIO_LSU_REG3_BYTE_COUNT,byte_count )|
                    CSL_FMK( SRIO_LSU_REG3_DRBLL_VAL, 0);

/***** Send the Data Out from Port#0, Generate Interrupt only for Error Condition ****/
SRIO_REGS->LSU[Lsuno].LSU_REG4 =   CSL_FMK( SRIO_LSU_REG4_OUTPORTID,0 )|
                    CSL_FMK( SRIO_LSU_REG4_PRIORITY,0 )|
                    CSL_FMK( SRIO_LSU_REG4_XAMBS,0 )|
                    CSL_FMK( SRIO_LSU_REG4_ID_SIZE,1 )|
                    CSL_FMK( SRIO_LSU_REG4_DESTID,0xBEEF )|
                    CSL_FMK( SRIO_LSU_REG4_INTERRUPT_REQ,1 )|
                    CSL_FMK( SRIO_LSU_REG4_SUP_GINT, 1)|
                    CSL_FMK( SRIO_LSU_REG4_SRCID_MAP, 0);

/*****************************************************************************************************************/
/***** STEP# 3: TRIGGER THE TRANSFER ****************************************************************************/
/*****************************************************************************************************************/

SRIO_REGS->LSU[Lsuno].LSU_REG5 =   CSL_FMK( SRIO_LSU_REG5_DRBLL_INFO,0x0000 )|
                    CSL_FMK( SRIO_LSU_REG5_HOP_COUNT,0x00 )|
                    CSL_FMK( SRIO_LSU_REG5_PACKET_TYPE,type );
```

At the end of the 3rd step, if it is certain that there is no other master going to use this LSU then there is no need to perform a check for the BUSY bit for the next LSU operation. Just check the FULL bit and lock the LSU register for writing to next set of shadow registers.

2.3.3 Message Passing

The communications port programming interface (CPPI) DMA module is the incoming and outgoing message-passing protocol engine of the RapidIO peripheral. Messages contain application-specific data that is pushed to the receiving device comparable to a streaming write. Messages do not contain read operations, but do have response packets.

With message passing, a destination address is not specified. Instead, a mailbox identifier is used within the RapidIO packet. The mailbox is controlled and mapped to memory by the local (destination) device. For multipacket messages, four mailbox locations are specified. Each mailbox can contain 4 separate transactions (or letters), effectively providing 16 destination addresses. Single-packet messages provide 64 mailboxes with 4 letters, effectively providing 256 destination addresses. Mailboxes can be defined for different data types or priorities. The advantage of message passing is that the source device does not require any knowledge of the destination device's memory map. The DSP contains buffer description tables for each mailbox. These tables define a memory map and pointers for each mailbox. The PKTDMA transfers messages to the appropriate memory locations via the DMA bus.

The ftype header field of the received RapidIO message packets is decoded by the logical layer of the peripheral. Only Type 11, Type 9, and Type 13 (transaction type 1) packets are routed to the PKTDMA module. Data is routed from the priority-based RX FIFOs to the CPPI module's data buffer within the shared buffer pool. The mbox (mailbox) header fields are examined by the mailbox mapper block. Based on mailbox, letter, source ID, and destination ID, the data is assigned the destination CPPI queue ID (QID) and flow ID. The maximum buffer space should accommodate 256 bytes of data, as that is the maximum payload size of a RapidIO packet. Each message in memory is represented by a buffer descriptor in the CPPI queue.

The SRIO message-passing functionality supports the following features.

- In-order message reception for dedicated flows is mode-programmable.
- Supports 16 segmentation contexts for receive and 16 segmentation contexts for transmit.
  - Allows tracking of simultaneous multisegment messages. A segmentation context is needed for each supported simultaneous multisegment RX or TX message.
  - RX segmentation contexts are used on first come, first serve basis.
- For Type 11 messages, one packet descriptor per message is placed on a CPPI queue to receive or transmit messages. This is desirable for inter-networking to eliminate gather and scatter operations. Type 9 operation can support multiple packet descriptors per message.
- Reception of out-of-order segments is allowed.
- Out-of-order responses are allowed.
- The transmit source must be able to RETRY any given segment of a transmitted message.
- An ERROR response is sent for the following conditions:
  - An incoming message can not be mapped to a CPPI destination queue using one of the programmable RX-mapping entries.
  - If a message is received for the local device, but the device is powered down (peripheral is only in packet-forwarding mode).
- A RETRY response is sent under the following conditions:
  - The targeted RX-free descriptor queue has no empty buffers (overflow).
  - The first segment of a multisegment message is received when all receive segmentation contexts are in use. Subsequent RETRY responses may have to be sent for received follow-on segments, until a segmentation context is available.

The following subsections describe more details of the message-passing operation.

2.3.3.1 RX Operation

The SRIO peripheral is capable of receiving incoming type 11 or type 9 RapidIO messages. It is also responsible for sending message responses to the original source device that sent a type 11 message. A response is sent for each segment of a Type 11 message. Responses have highest system priority. The SRIO peripheral generates a response at a priority level of +1 from the corresponding request. Additionally, the CRF bit can be set on out-going message responses. The priority can be promoted further to overcome times of congestion based on the Promote_Dis bit in the RIO_PER_SET_CNTL register.

2.3.3.1.1 RX Message Storage

The RX functionality in the SRIO peripheral tracks all the message segments and uses the PKTDMA to correctly reassemble the message payloads within memory. Using CPPI packet descriptors with the queue manage and the PKTDMA provides a scheme for tracking single and multipacket messages, queuing messages for the receiving entities on completion, and generating interrupts. In addition to the payload, it also provides a way to communicate SRIO packet header info to and from the DSP cores. The queue manager subsystem supports interrupt generation for the data destination, as well as interrupt-pacing mechanisms based on message received counts and elapsed time.

For each segmentation context, there are resources to track reception of all the message segments for a given type 11 message. There is also a receive segment timer which limits the length of time between receiving two adjacent segments of a message. If this timer is exceeded, the hardware notifies the PKTDMA to close the context. If the new segment of the message does eventually come after the timer is expired, a new context is opened for the message. If all RX segmentation contexts are in use, and another type 11 message is received, the peripheral sends RETRY responses to the sending device until a segmentation context is open. If a type 9 message is received while no contexts are available, the peripheral drops the message.

2.3.3.1.2 RX Message Mapping

The RX message-mapping functionality in the SRIO peripheral directs the inbound messages to the appropriate FlowID, and can also override the destination queue specified in the FlowID configuration. The mapping is programmable and must be configured after device reset. Typically, the received messages are placed on the destination queues designated to the CPU or packet accelerator once the entire message is received.

As a message segment is received, the following steps are taken.

- Type11: Based on SRCID, DESTID, MBOX, and Letter map the message to a FlowId. The mapping of the fields to a FlowId is done through the RIO_RXU_MAPxx_L/H and RIO_RXU_MAPxx_QID registers.
- Type9: Based on SRCID, DESTID, STREAMID, COS, PRIO, CRF map the message to a FlowId. The mapping of the fields to a FlowId is done through the RIO_RXU_TYPE9_MAPxx_L/H and RIO_RXU_MAPxx_QID registers.

There are 64 programmable mapping entries. Each mapping entry consists of three registers. Type 11 messages use the RIO_RXU_MAPxx_L, RIO_RXU_MAPxx_H, and RIO_RXU_MAPxx_QID registers while Type 9 messages use the RIO_RXU_TYPE9_MAPxx_L, RIO_RXU_TYPE9_MAPxx_H, and RIO_RXU_MAPxx_QID registers. The first two registers of each mapping entry provide the criteria to match an incoming message, while the third specifies the FlowID, and optionally the destination queue ID. Both type 11 and type 9 use the same RIO_RXU_MAPxx_QID registers. The details of these registers are found in *Chapter 3*.

The mapping entries can be programmed for specific accesses (such as a given mailbox/letter, or COS/streamID, sourceID, or DestID), or can provide more general access using the mask and promiscuous fields. For example, the mask fields can be used to grant multiple mailbox/letter combinations access to a queue using the same table entry. A masking value of 0 in the mailbox or letter mask fields indicates that the corresponding bit in the mailbox or letter field is not used to match for this queue mapping entry. A mailbox mask of all zeros would allow a mapping entry to be used for all incoming mailboxes. Similarly, because the RapidIO peripheral can support an operational mode which allows multiple device IDs, the mapping registers can be assigned to unique destIDs if desired.

Another feature of the mapping table entry is to provide a security feature to enable or disable access from specific external devices to local mailboxes. The SOURCEID field is used to indicate which external device has access to the mapping entry and corresponding CPPI queue. A compare is performed between the sourceID of the incoming message packet and each relevant mailbox/letter table mapping entry SOURCEID field. A PROMISCUOUS bit allows this security feature to be disabled. When the PROMISCUOUS bit is set to 1, full access to the mapping entry from any SOURCEID is allowed. Note that when the PROMISCUOUS bit is set, the mailbox/letter and corresponding mask bits are still in effect. When the PROMISCUOUS bit is 0, it is equivalent to a mask value of 0xFFFF, and only the matching SOURCEID is allowed access to the mailbox.

Each table entry also indicates if it used for single or multi-segment message mapping. Single-segment message mapping entries utilize all six bits of the mailbox and corresponding mask fields. Multi-segment uses only the 2 LSBs.

If an incoming message does not match at least one mapping entry, the message is discarded and an ERROR response sent. Additionally, the transaction is logged in the logical layer error management capture registers and an interrupt is set. When an incoming message is compared against the mapping entries, it may match more than one mapping entry, and uses the first matching entry of the 64 available.

After a message is mapped properly, the FlowID identifies which buffer and descriptor queues should be used for storing the payload. The PKTDMA can pull a new packet descriptor from one of the free descriptor queues based on the FlowID. Each memory region of the device (individual L2s, shared memory, and DDR) can support four free descriptor queues.

If the RX message's calculated length is greater than the available free descriptor queues offer, the PKTDMA will use multiple buffer descriptors (scattered payload buffers). Type 9 does not have an issue with this model. However, for type 11, a RETRY may result in nondeterministic behavior, such as out-of-order segments in the buffers, over-writing of segments within a buffer, or non-contiguous payload within a buffer.

The destination queue information is passed to the PKTDMA. Once the PKTDMA has the entire payload for a transaction, it transfers the data from the queue manager to the relevant destination queue. The software must ensure that the biggest defined queue size should be big enough to accept the maximum size payload, plus any control packets, in a single buffer. TI also recommends that the user choose the last buffer selection out of the four for a region as a DDR buffer descriptor. Thus, if the local memory is full, the data can be stored in a DDR.

Usable Multiple Free Queues

Although the CDMA allows for up to four free descriptor queues per FlowID, the SRIO peripheral will most likely make use of only two queues for type 11 messages. The reason for this is that the CDMA attached to the SRIO peripheral needs an exact byte count from the SRIO peripheral when handing it a new message, as it puts this exact message size in the descriptor. It cannot use ssize * segment for multi-segment messages to give the CDMA a worst case number, which is the best the SRIO peripheral could do as the last segment of the message can be smaller than the ssize. The SRIO peripheral does know the exact size of single segment messages and can pass this info.

For multi-segment messages, the SRIO peripheral passes an Unknown status for the size of the message to the CDMA. The CDMA then keeps a running tally as the message segments are passed to it. When the message is completely received, the total size is written to the descriptor by the CDMA.

Due to the nature of SRIO RETRY and out-of-order reception of message segments, a type 11 message must use only one descriptor/buffer. Behavior of the RXU is undefined if this is not the case. The SRIO only support host packet descriptor types.

The only solution for multi-segment type 11 messages is to support the biggest size message in the system, which in almost all cases will be 4KB. Type 9 traffic is slightly different. Again, it sends the Unknown size to the CDMA; however, type 9 messages can use multiple buffers/descriptors per message, which will be linked together. This is because there is no out-of-order issues to deal with.

Descriptor Dry Out

The RXU and CDMA combination only check for the dry-out condition when a new context is opened. It does not check when all contexts are in use (RETRY). If there is either a single-segment message, or a small multi-segment message that can be completely buffered in the CDMA buffers (and it detects or runs into the starvation condition) such that the context is reopened, and a new message comes in that uses that context, then that new message will also be retried.

If the dry-out issue occurs on a multi-segment message that can't fit into the CDMA buffers, then the RXU is stalled. When the RXU is stalled, all channels are stalled, not just the one channel with the dry-out issue. If the condition lasts long enough, such that all the RX shared buffers of the logical layer are used, then other protocol units such as the TXU, LSU, MAU traffic are also blocked.

The key is to ensure the dry-out issue never occurs. The CDMA does not have the ability to move to a different descriptor queue if the dry-out issue is detected on the original targeted one. The only feasible solution is to take advantage of the return push policy for the descriptors, which allows the free descriptors to push either the head or tail of the queue. This way, descriptors from one queue can be in two memory areas.

For example, L2 descriptors can be at the head of the queue, which are the main descriptors in use. These descriptors are used first and when returned to the free queue, they return to the head of the queue. DDR descriptors, primarily for overflow situations and preventing dry-out conditions, are in the same queue. If these descriptors are used again, then they will return to the tail of the free queue. The return policy is governed by the Return Push Policy bit (14) of the host packet descriptor word 2, and must use the return queue specified inside the descriptor.

Out of Order Handling (only for type 11)

In the case of a RETRY situation (resulting from lack of segmentation context or packet descriptor availability), it is possible that RX resources become available during the arrival of non-SOP segments of that message. This results in acceptance of out-of-order segments. To manage this, the SRIO peripheral passes buffer offset information along with a ThreadID for each segment to the PKTDMA so that the CPPI buffer reflects a completely in-order message.

Zero bytecount

If there is an EOP along with a 0 bytecount segment, it terminates the PDU and immediately closes the segmentation context.

If the zero bytecount is not accompanied by an EOP, this implies that there are followup segments. Thus, the context will remain open. If there were none to be received, there will eventually be a timeout.

2.3.3.1.3 RX Protocol-Specific Descriptor Information

The PKTDMA creates the RX host packet descriptors as specified in the CPPI v4.2 specification. These descriptors are used to point to corresponding data buffers in memory. Additionally, descriptors hold protocol-specific information that can be used to construct or deconstruct packets, and help route packets to the proper destination. RapidIO packet header information is stored in the packet descriptors. The header information is passed by the RXU across the CPPI FIFO streaming interface and populated into the packet descriptors by the PKTDMA. As segments of a received message arrive, the msgseg field of each segment is monitored to detect the completion of the received message. Once a full message is received, the descriptor is completed by the PKTDMA and the queue manager subsystem delivers the descriptor to the destination queue. Figure 2-18 shows the host packet descriptor RapidIO protocol specific information.

RapidIO packet descriptors are a contiguous block of at least 10 32-bit data words aligned on a 16B boundary. Accesses to these registers are restricted to 32-bit boundaries. The offset in Figure 2-18 is with respect to the protocol-specific words in the CPPI specification. The PKTDMA must construct these descriptors based on the side-band signals sent by the RXU. The RXU sends the protocol-specific descriptor fields as status words to the PKTDMA, after the entire payload has been sent to the PKTDMA.

Figure 2-18. RX RapidIO Protocol Specific Descriptor Fields (Type 11)

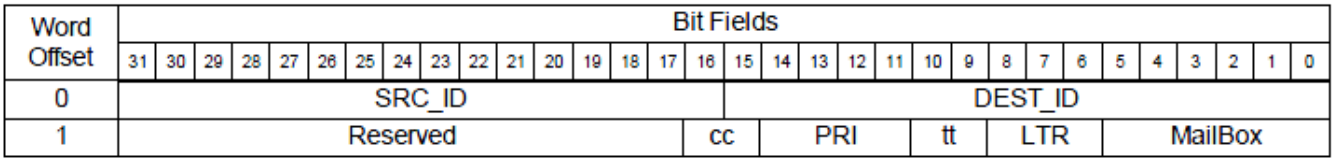


Table 2-16. RX RapidIO Protocol Specific Packet Descriptor Fields (Type 11)

Field	Description
Src_ID	Source Node ID — Unique node identifier of the source of the message.
Dest_ID	Destination Node ID the message was sent to.
tt	RapidIO *tt* field specifying 8- or 16-bit deviceIDs 00h = 8b deviceIDs 01h = 16b deviceIDs 10h = Reserved 11h = Reserved
PRI	Message Priority — Specifies the SRIO priority at which the message was sent. VC||PRIO||CRF
CC	Completion Code: 00h = Good Completion. Message received. 01h = Teardown 10h = Error, TimeOut on receiving one of the segments 11h = Length mismatch between size in the UDI packet and received payload
LTR	Destination Letter — Specifies the letter to which the message was sent. For the encoding of 0b100 below, the hardware checks for an unused context starting with letter = 0, and incrementing to letter = 3. The first unused context with that letter will be used. If there are no context available with any letters then the packet is stopped and re-arbitrated in the TXU until one does become available. 0b000 — Letter 0 … 0b011 — Letter 3 0b100 — First available. Assigned by the hardware. 0b1xx — Reserved.
Mailbox	Destination Mailbox — Specifies the mailbox to which the message was sent. 0b000000 — Mailbox 0 0b000001 — Mailbox 1 … 0b000100 — Mailbox 4 … 0b111111 — Mailbox 63

Table 2-17. RX RapidIO Protocol Specific Descriptor Fields (Type 9)

Word Offset	Bit Fields																															
	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
0	SRC_ID																DEST_ID															
1	Reserved															CC	PRI			tt	LTR			Mailbox								

Table 2-18. RX RapidIO Protocol Specific Packet Descriptor Fields (Type 9)

Field	Description
Src_ID	Source Node ID – Unique node identifier of the source of the message. If the tt indicates an 8b device ID, the upper 8 bits of this field must be 8'b0
Dest_ID	Destination Node ID the message was sent to. If the tt indicates an 8b device ID, the upper 8 bits of this field must be 8'b0
STRM_ID	Stream ID
PRI	Message Priority – Specifies the SRIO priority at which the message was sent. VC||PRIO||CRF
CC	Completion Code: 00h = Good Completion. Message received. 01h = Teardown 10h = Error, TimeOut on receiving one of the segments 11h = Length mismatch between size in the UDI packet and received payload
T	This is TT [0] bit only. RapidIO *tt* field specifying 8- or 16-bit deviceIDs 0h = 8b deviceIDs 1h = 16b deviceIDs
COS	Class of Service
R	Reserved

2.3.3.1.4 RX Error Handling

Table 2-19lists situations handled by the SRIO with the relevant error code.

Table 2-19. Error Codes

Error Condition	Action	CC	Type9/11
The packet comes from the UDI, but the shared buffer is full	Retry		11
The data is sitting in the shared buffer, but when opening a context for it, there is no mapping table that matches the descriptor.	Error		11
A context is open, however the following segment timed out.	Close context	010	9,11
Total length described in the descriptor does not match the actual payload that came with the message.	No action	"011"	9,11

2.3.3.1.5 RX Teardown

A CPU may wish to stop receiving messages and reclaim buffers belonging to a specific channel. This is called RX queue teardown. When the host desires to teardown a channel, it writes a 1 to the teardown field of the corresponding RX Channel Global Configuration Register for that channel. The host must not disable the channel by writing a 0 to the enable field of this register. The PKTDMA does this when finished with the teardown. When the PKTDMA detects the teardown bit is set, it will:

- Signal the attached peripheral (rx_teardown_req) that a teardown was requested by the host.
- Wait for the peripheral to acknowledge (rx_teardown_ack).
- Notes:
  - During this time, processing continues normally.
  - In some peripherals, rx_teardown_req and rx_teardown_ack are tied together in loopback.
- Following ack, the current packet is allowed to complete normally.
- Clear the channel enable in the RX Channel Global Configuration Register. The teardown bit remains set as an indication that a teardown was performed.

2.3.3.2 TX Operation

The TXU is the logical layer protocol unit responsible for transmitting type 11 RapidIO messages as well as type 9 data streaming packets. It supports the following:

- It transmits payload data setup in data buffers, pointed to by the PKTDMA descriptors, as message segments.
- Type 11:
  - Every outgoing message segment gets a response packet back. It is responsible for receiving message responses for every message segment sent out.
  - In case of a RETRY response, it retransmits the specific segment on which the retry response came
  - Once responses for all message segments are received, it also releases the corresponding descriptor.
- Type 9: No response is expected. Once a segment is sent, the TXU is finished with it.

A TX PKTDMA is inside the SRIO subsystem. It has the following features:

- The PKTDMA has one TX prefetch FIFO per queue.
- The PKTDMA talks to the queue manager, and accesses the descriptors and stores them in its internal prefetch buffers. It is not responsible for releasing the descriptor once it sends the data to SRIO. Once all good completion response packets are received for a payload, the SRIO indicates to the PKTDMA that it can release the buffer descriptor for type11. The SRIO send the descriptor information across to the PKTDMA. For type9, the SRIO asks for the descriptor to be released once it gets all the payload as there is no response packet to wait for.
- Fetching payload from data buffers into its internal prefetch buffers. The latency of fetching the payload to the prefetch buffers is non-deterministic.
- Interfacing with the TXU scheduler in the following ways:
  - It notifies the TXU of the total data available
  - It notifies the TXU if it there is at least one EOP in its buffer space
  - It transfers the data and descriptor information to the TXU when requested

The PKTDMA has registers that fix the priority of each queue. More than one queue could be programmed to be at the same priority. Inside the TXU, each queue is programmed to transmit data to a fixed port at a specific CRF value. The TXU additionally gets the priority information from the PKTDMA. The sequence of operation is as follows.

1. An external master, which could be a CPU core or a packet accelerator; for example, first writes the payload into the data buffers.
2. Same master sets up the CPPI packet descriptors for a specific queue.
3. The PKTDMA pre-fetches the packet descriptor and then the payload information and stores it in the pre-fetch FIFO. Once the data is available in this FIFO, the PKTDMA informs the TXU.
4. Based on the various queues that have data available, the TXU requests the PKTDMA to send data. The first thing the TXU receives from the PKTDMA are the protocol-specific packets. Once the credit manager gives the TXU a credit to transmit the data, based on the information in the protocol-specific packets, the TXU gets the payload from the queue with the highest priority.
5. The data is then sent to the TX-shared buffer and sent out through the UDI interface to the ports. Figure 2-19 shows a high-level architecture for the TX CPPI scheme for SRIO.
6. On type 9, when the PKTDMA sends the EOP on the message, the TXU asks the PKTDMA to close the descriptor immediately by sending the descriptor index to be released, as well as the queue it wants the descriptor to be sent to. The PKTDMA then closes the descriptor.
7. For type 11, the response packet comes back on the UDI interface.
8. The TXU checks the response packet and keeps track of all the segments and the responses in the CAM. If the response is retry, the TXU initiates a RETRY for the message segment through the VBUSM interface in the CAU. If the response is an error, the PKTDMA is requested to send the descriptor to one of the garbage queues when the entire message is sent and all responses received. If the response is good, the segment is checked off in the CAM. When all segments get a good response, the PKTDMA is asked to free the descriptor.

2.3.3.2.1 TX Protocol-Specific Descriptor Information

A TX RapidIO packet descriptor is a contiguous block of at least 10 32-bit data words aligned on 16B boundaries. There is a single packet descriptor per RapidIO message. The descriptor information is split as follows:

- Words 0-7 of the descriptor hold the control information such as data buffer start address, and total message payload byte count. This information is sent to the TXU as sideband signals.
- TX CPPI-specific information is stored in the protocol-specific section of the descriptor. If there are no other information packets between the control packets [0-7], then these would be in packets 8 and 9. However, there is no such restriction imposed on the descriptor. The details of the protocol-specific descriptor are shown in Figure 2-19. The word offset is with respect to the protocol-specific words of a host packet descriptor.

The transmit message length field (programmed in the non-RapidIO-specific TX packet descriptors fields) must be a double-word multiple, as required by the RapidIO messaging specification. The destination port number, as well the priority (VC||PRIO||CRF), is implied by the queue from which the data is coming to the TXU by using the RIO_TX_QUEUE_SCH_INFOx registers.

Figure 2-19. TX RapidIO Protocol-Specific Descriptor Fields

Word Offset	Bit Fields																													
	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
0	SRC_ID																DEST_ID															
1	Rsv								Retry_Count				SSIZE				Rsvd						tt		LTR			MailBox				

Table 2-20. TX RapidIO Protocol-Specific Packet Descriptor Field Definitions

Field	Description
Src_ID	Source Node ID — Unique node identifier of the source of the message.
Dest_ID	Destination Node ID the message was sent to.
Retry_Count	Message Retry Count — Set by the CPU to indicate the total number of retries allowed for this message, including all segments. Decremented by the port every time a message is retried. • 0b000000 — Infinite retries • 0b000001 — Retry message one time • 0b000002 — Retry message two times • … • 0b111111 — Retry message 63 times
SSIZE	RIO standard message payload size. Indicates how the hardware should segment the outgoing message by specifying the maximum number of bytes per segment. If the message is a multisegment message, this field remains the same for all outgoing segments. All segments of the message, except for the last segment, have payloads equal to this size. The last message segment may be equal or less than this size. Maximum message size for a 16-segment message is shown below. *Message_length/16 must be less than or equal to Ssize*; if not, the message is not sent and the descriptor is sent to a Garbage_QID_SSIZE. • 0b0000 - 0b1000 — Reserved • 0b1001 — 8-byte message segments (Supports up to a 128B message with 16 segments) • 0b1010 — 16-byte message segments (Supports up to a 256B message with 16 segments) • 0b1011 — 32-byte message segments (Supports up to a 512B message with 16 segments) • 0b1100 — 64-byte message segments (Supports up to a 1024B message with 16 segments) • 0b1101 — 128-byte message segments (Supports up to a 2048B message with 16 segments) • 0b1110 — 256-byte message segments (Supports up to a 4096B message with 16 segments) • 0b1111 — Reserved
TT	RapidIO *tt* field specifying 8- or 16-bit deviceIDs • 00 — 8b deviceIDs • 01 — 16b deviceIDs • 10 — Reserved • 11 — Reserved

Table 2-20. TX RapidIO Protocol-Specific Packet Descriptor Field Definitions (continued)

Field	Description
LTR	Destination Letter — Specifies the letter to which the message was sent. • 0b000 — Letter 0 • … • 0b011 — Letter 3 • 0b1xx — Reserved
Mailbox	Destination Mailbox — Specifies the mailbox to which the message was sent. • 0b000000 — Mailbox 0 • 0b000001 — Mailbox 1 • … • 0b000100 — Mailbox 4 • … • 0b111111 — Mailbox 63

Type 9

The transmit message length field cannot exceed 64KB. The priority, CRF, and VC information for the transaction is received from RIO_TX_Queue_Scheduler_Info[1-4] registers. The destination port is also based on the RIO_TX_Queue_Scheduler_Info[1-4] registers. The segment size is based on the MTU field of the CAR register. The minimum payload can be 1 byte; however it is padded in that case, to make the minimum PDU of one double word. The length of the entire payload is sent by the PKTDMA before sending the payload itself.

Figure 2-20. TX RapidIO Protocol-Specific Descriptor Fields Type 9

Word Offset0	SRCID [31:16]			DEST_ID[15:0]	
Word Offset1	StreamID [31:16]	Reserved [15:11]	TT[10]	Reserved [9:8]	COS [7:0]

Table 2-21. TX RapidIO Protocol-Specific Packet Descriptor Field Definitions

Name	Description
Src_ID	Source Node Id — Unique node identifier of the source of the message.
Dest_ID	Destination Node ID the message was sent to.
COS	Class of Service. This field is used as one of the qualifications before opening up a new context
TT	RapidIO *tt* field specifying 8- or 16-bit deviceIDs • 00 — 8b deviceIDs • 01 — 16b deviceIDs • 10 — Reserved • 11 — Reserved
StreamID	Stream ID for the transaction. This would be sent out in the first packet to the UDI interface; however, it is suppressed in the following packets for the same transaction.

2.3.3.2.2 TX Scheduler

The scheduler is more deterministic than the legacy weighted round-robin method supported on earlier devices. The 16 TX queues are organized by output port number and priority. Priorities include the CRF bit in addition to the 2 normal priority bits, which allows 8 priorities for the scheduler to consider. The VC bit is currently not used. The port and priority assigned to each TX queue are programmed in the TX_QUEUE_SCH_INFO1/2/3/4 registers. These registers can only be written to while the peripheral is disabled. They are common to type 9 and type 11 traffic. The details of the registers are found in Chapter 3.

The scheduler looks at the descriptor information for each queue with data in it and qualifies it as follows:

- Packet descriptor error check: The CPPI-specific packet descriptor information and payload byte count information is checked to make sure there are no errors. For example, the payload byte count must be double-word aligned, and the payload byte count must be smaller than SSIZE × 16. If this is not the case, packet transmission is avoided and the descriptor is returned to a garbage collection queue rather than being returned to the free descriptor queue. If the descriptor information is legitimate but the SSIZE has been programmed to something less than 256B, the peripheral is responsible for breaking up the sent data into multiple RapidIO packets.
- XOff Check: The DestId is checked against the TX_CPPI_Flow_Mask register to determine which queues are available for transmission and which are flow controlled (see congestion control). If the queue has been flow-controlled, it is temporarily disallowed, and thus because it is not part of the scheduling algorithm it is stalled, resulting in HOL blocking for that queue. If it is not flow-controlled, the queue can be used by the scheduler algorithm.
- Unique mailbox, letter, SrcID, DestID check: The next check is for a given source and destination pair. Only one unique mailbox and letter combination can be outstanding at any given time to a specific SrcID/DestID pair. A new message is temporarily ignored if it conflicts with an existing mailbox, letter, SrcID, and DestID combination which is already in use, even if the new message is at a different priority, until a completion code (CC) is received on the preceding transaction.

Once it is determined which queues are available, the scheduler searches only the highest priority queues. It sends this priority and port information to the credit manager, which then grants it credit based on its own qualifications. If multiple queues qualify for transmit at the same priority, the scheduler will simply round-robin between those queues. Thus it sends all packets of higher priority queues before moving on to lower priority queues of a specific port. If two different ports have different priorities, the scheduler will round-robin between them to ensure that a port is not starved. The CRF bit gives preferential treatment to those queues programmed without the CRF bit. The scheduling algorithm interleaves TX messages from different queues on a segment basis. For example, if 4 TX queues are assigned to priority 2 (one on each outbound port) and all these queues are not empty, the scheduler sends one segment from each queue before moving to the second segment of any queue. Credits are based on a maximum size (256B) payload. If the SSIZE is less than 256B, each packet may occupy only a part of the 256B in the TX Shared FIFO. Transactions from a given queue are guaranteed to be in order.

2.3.3.2.3 TX Response Handling

All out-going message segments have responses that indicate the status of the transaction. Responses may indicate DONE, ERROR, or RETRY. After all the responses for each segment are received for a packet descriptor, the TXU must release the descriptor.

The release mechanism of the descriptor is dependent on the response received:

- DONE: If all segments come back with a response of DONE, the scheduler requests the PKTDMA to release the specific descriptor. It sends the descriptor details to the PKTDMA.
- ERROR: The descriptor is written to a garbage collection queue rather than being returned to the free descriptor queue. There are garbage queues for each error type supported. Based on the garbage queue that the software is reading from, it will the exact error that occurred. The address of the garbage collection queue is written in the garbage collection queue number register shown in Table 3-61. A designated CPU services the garbage collection queues.
- RETRY: As soon as the RETRY response is received, the TXU issues a read for the appropriate size through the VBUSM port in the CAU. When the PKTDMA starts sending the message for the first time, it also sends the base address from which the payload must be transferred (as a side band signal). Thus, to calculate the exact address from which the RETRY has to be issued, the TXU uses the original base address, SSIZE, as well as the SRCTID that comes back in the response, to calculate the address from which the RETRY has to be done. On a retry, the data is received through the CAU VBUSM payload bits, not the PKTDMA payload bits. Retry of a message segment does not imply retrying a whole message. Only segments for which a RETRY response is received should be re-transmitted. If the amount of RETRY exceeds the RETRY_COUNT (set by the user), the descriptor, attempting to be transmitted, is sent to the retry garbage queue.

Because RapidIO allows for out-of-order responses, the TXU hardware must support this functionality. The actual transmission order of the messages is not relevant, as only the order of the completed responses determines when the packet descriptors are freed. This limits the SRIO to use only a single data buffer descriptor through the PKTDMA. Unfortunately, buffer linking is not possible. To support greater performance, it is not necesary to wait for a descriptor to be freed before proceeding to transmit the next descriptor in that queue.

A transaction timeout is used by all outgoing message segments. It is defined by the 24-bit value in the port response time-out CSR.

2.3.3.3 Message Passing Software Requirements

The main software requirements for message passing are to set up the previously mentioned RX message-mapping registers and TX scheduler (port and priority information) registers.

Some applications require in-order delivery of messages. This requirement is complicated by the fact that message retries can exist on any given message segment. The only true way to ensure in-order message delivery is to explicitly use the same mailbox and letter combination between two endpoints using the same deviceIDs. This ensures in-order delivery (barring any error conditions) as a message is sent out only after the previous one has completed. This is managed by the CAM functionality discussed earlier. Performance is impacted using this method because there can not be more that one message in-flight at a given time.

Higher performance in-order DSP-to-DSP delivery is possible under the following restrictions. Because the TX queues are based on priority, reordering is not done by the fabric or peripheral's physical layer, thus in-order delivery to a given DestID can be done by using the same TX queue even when the first available letter is used. In this case, only logical layer retries can cause messages to be received out of order.

The following must be obeyed to prevent the occurrence of logical layer retries:

- Ensure that the RX free descriptor queues are never empty.
- Do not exceed the limitation of 16 open segmentation contexts in the RXU.

The hardware does not have to do anything special to enable this. This is in control of the user.

2.3.3.3.1 Disable the TX and RX SRIO Channels

```
enable_tx_chan(SRIO_CDMA_TX_CHAN_REGION, CHANNEL0, DISABLE);

for (idx = 0; idx < 16; idx++)
     enable_rx_chan(SRIO_CDMA_RX_CHAN_REGION, idx, DISABLE);

Please refer Appendix: A for more details about these function calls.

CREATE HOST PACKET DESCRIPTORS FOR TX OPERATION FOR TX QUEUE#672

/***** POP THE DESCRIPTOR # 1 *****/
host_desc[0]    = pop_queue(HOST_TX_COMPLETE_Q);

/***** INITIALIZE THE GENERAL PART OF THE DESCRIPTOR # 1 *****/

// CPPI_HostPacketDescriptor is the Host Descriptor Structure Type
host_pkt    = (CPPI_HostPacketDescriptor *)host_desc[0];

// Protocol Specific Words are located in the Descriptor
host_pkt->ps_reg_loc  = 0;

// SRIO Peripheral has 2 words of Protocol Specific Descriptor Words
host_pkt->psv_word_count  = 2;

// TX_PACKET_LENGTH is the length of the Transmit Packet
host_pkt->packet_length  = TX_PACKET_LENGTH;

// Descriptor has just two buffers so link it to next buffer
host_pkt->next_desc_ptr  = host_desc[1];
```

```
host_pkt->src_tag_lo = 0;

// TX_PAYLOAD0_PTR is the Pointer to the Tx Payload
host_pkt-
>buffer_ptr  = TX_PAYLOAD0_PTR;     /***** INITIALIZE THE PROTOCOL SPECIFIC PART OF THE
DESCRIPTOR # 1 *****/

// Point after 8 words to the End of the Descriptor For Protocol Specific Part
temp    = (Uint32 *) (host_pkt + 1);

// Source Id=0x1234, Destination Id=0x5678
temp[0]    = 0x12345678;

// tt=1, Letter=1, Mail Box=2
temp[1]    = 0x00000242;


/***** POP THE DESCRIPTOR # 2 *****/
host_desc[1]    = pop_queue(HOST_TX_COMPLETE_Q);


/***** INITIALIZE THE GENERAL PART OF THE DESCRIPTOR # 2 *****/

// CPPI_HostPacketDescriptor is the Host Descriptor Structure Type
host_pkt    = (CPPI_HostPacketDescriptor *)host_desc[1];

// TX_PACKET_LENGTH is the length of the Transmit Packet
host_pkt->packet_length  = TX_PACKET_LENGTH;

// Descriptor has just two buffers so link it to next buffer
host_pkt->next_desc_ptr  = NULL;
host_pkt->src_tag_lo = 0;

// TX_PAYLOAD1_PTR is the Pointer to the Tx Payload
host_pkt->buffer_ptr  = TX_PAYLOAD1_PTR;


CONFIGURE THE TX QUEUES & TX CHANNELS

// Set the Tx queue threshold to be 1 for Queue#672
set_queue_threshold (672, 0x81);

// Disable & then configure the Tx channel#0
config_tx_chan (SRIO_CDMA_TX_CHAN_REGION, CHANNEL0, HOST_TX_COMPLETE_Q);

// Configure the Tx Channel#0 Priority to be High
config_tx_sched(SRIO_CDMA_TX_SCHD_REGION, CHANNEL0, 0);
```

2.3.3.3.2 Configure the RX Channel Flows

```
/***** Create flow configuration 0 for the Host packets *****/

// Rx descriptor is Host type and having protocol specific fields. Rx destination Queue is 900.
flow_a = 0x24000000 + 900;

// Configure the Host Rx Free Descriptor Queue for Descriptors to be Queue#2000
flow_d =(HOST_RX_FDQ << 16) + HOST_RX_FDQ;
flow_e = flow_d;

// Write above configured values to the CDMA Channel Flow#0
config_rx_flow(SRIO_CDMA_RX_FLOW_REGION, FLOW0,
                   flow_a, 0, 0, flow_d, flow_e, 0, 0, 0);
```

2.3.3.3.3 Configure the RX Mapping Table

```
// Host Message #1
value = 0x3    << 30   // LTR_MASK
          | 0x3F   << 24   // MBX_MASK
          | 1      << 22   // LTR
          | 2      << 16   // MBX
          | 0x1234 << 0;   // SRCID
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_L0) + (0 * 3);
*reg = value;

value = 0x5678 << 16   // DESTID
          | 0      << 15   // DEST_PROM
          | 1      << 13   // TT
          | 0      << 1     // SRC_PROM
          | 0      << 0;   // SEG_MAP
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_H0) + (0 * 3);
*reg = value;

value = 0      << 16   // FLOWID
         | 900    << 0;   // DEST_QID
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_QID0) + (0 * 3);
*reg = value;


// Host Message #2
value = 0x3    << 30   // LTR_MASK
          | 0x3F   << 24   // MBX_MASK
          | 3      << 22   // LTR
          | 7      << 16   // MBX
          | 0x9abc << 0;   // SRCID
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_L0) + (1 * 3);
*reg = value;

value = 0xdef0 << 16   // DESTID
          | 0      << 15   // DEST_PROM
          | 1      << 13   // TT
          | 0      << 1     // SRC_PROM
          | 0      << 0;   // SEG_MAP
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_H0) + (1 * 3);
*reg = value;

value = 0      << 16   // FLOWID
         | 900    << 0;   // DEST_QID
reg = ((Uint32 *)&srioRegs->RIO_RXU_MAP_QID0) + (1 * 3);
*reg = value;
```

2.3.3.3.4 Configure TX Port and Priority Registers

```
// Configure TX port/priority registers
value = 0    << 28   // Queue 3 Port (0 - 3)
      | 0    << 24   // Queue 3 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 20   // Queue 2 Port (0 - 3)
      | 0    << 16   // Queue 2 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 12   // Queue 1 Port (0 - 3)
      | 0    <<  8   // Queue 1 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    <<  4   // Queue 0 Port (0 - 3)
      | 0    <<  0;  // Queue 0 Priority (0 - 7) (includes PRIO bits and CRF bit)
reg = ((Uint32 *)&srioRegs->RIO_TX_QUEUE_SCHEDULER_INFO0) + 0;
*reg = value;

value = 0    << 28   // Queue 7 Port (0 - 3)
      | 0    << 24   // Queue 7 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 20   // Queue 6 Port (0 - 3)
      | 0    << 16   // Queue 6 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 12   // Queue 5 Port (0 - 3)
      | 0    <<  8   // Queue 5 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    <<  4   // Queue 4 Port (0 - 3)
      | 0    <<  0;  // Queue 4 Priority (0 - 7) (includes PRIO bits and CRF bit)
reg = ((Uint32 *)&srioRegs->RIO_TX_QUEUE_SCHEDULER_INFO0) + 1;
*reg = value;

value = 0    << 28   // Queue 11 Port (0 - 3)
      | 0    << 24   // Queue 11 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 20   // Queue 10 Port (0 - 3)
      | 0    << 16   // Queue 10 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 12   // Queue 9 Port (0 - 3)
      | 0    <<  8   // Queue 9 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    <<  4   // Queue 8 Port (0 - 3)
      | 0    <<  0;  // Queue 8 Priority (0 - 7) (includes PRIO bits and CRF bit)
reg = ((Uint32 *)&srioRegs->RIO_TX_QUEUE_SCHEDULER_INFO0) + 2;
*reg = value;

value = 0    << 28   // Queue 15 Port (0 - 3)
      | 0    << 24   // Queue 15 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 20   // Queue 14 Port (0 - 3)
      | 0    << 16   // Queue 14 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    << 12   // Queue 13 Port (0 - 3)
      | 0    <<  8   // Queue 13 Priority (0 - 7) (includes PRIO bits and CRF bit)
      | 0    <<  4   // Queue 12 Port (0 - 3)
      | 0    <<  0;  // Queue 12 Priority (0 - 7) (includes PRIO bits and CRF bit)
reg = ((Uint32 *)&srioRegs->RIO_TX_QUEUE_SCHEDULER_INFO0) + 3;
```

2.3.3.3.5 Enable TX and RX Channels

```
enable_tx_chan(SRIO_CDMA_TX_CHAN_REGION, CHANNEL0, ENABLE);

for (idx = 0; idx < 16; idx++)
     enable_rx_chan(SRIO_CDMA_RX_CHAN_REGION, idx, ENABLE);



TRIGGER THE TX OPERATION

// Trigger the Tx Operation by Pushing the Tx Descriptor in the Queue#672
push_queue(672, 1, 0, host_desc[0]);


CHECK WHETHER TX OPERATION COMPLETE OR NOT

while (hostRxCount < 1)
{
       // Get current descriptor count for host RX destination queue#900
       value = get_descriptor_count(900);
       hostRxCount = value;
}
```

NOTE: Although it is advisable to enable all the channels in RX, enabling a subset of channels also works.

2.3.4 Maintenance

The type 8 MAINTENANCE packet format accesses the RapidIO capability registers (CARs), command and status registers (CSRs), and data structures. Unlike other request formats, the type 8 packet format serves as both the request and the response format for maintenance operations. Type 8 packets contain no addresses, and only contain data payloads for write requests and read responses. All configuration register read accesses are word (4-byte) accesses. All configuration register write accesses are also word (4-byte) accesses.

The WRSIZE field specifies the maximum size of the data payload for multiple double-word transactions. The data payload may not exceed that size but may be smaller if desired. Both the maintenance read and the maintenance write request generate the appropriate maintenance response.

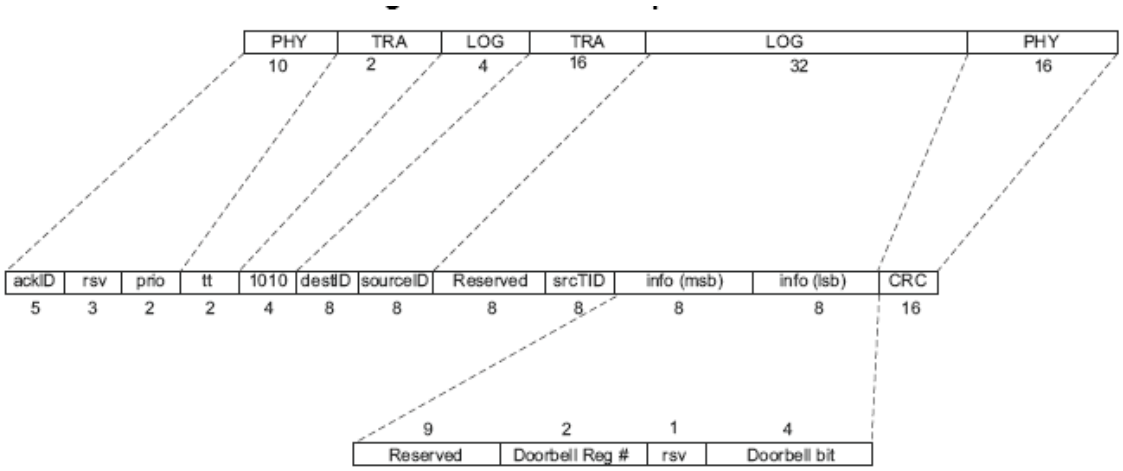
The maintenance port-write operation is a write operation that does not have guaranteed delivery or an associated response. This maintenance operation is useful for sending messages such as error indicators or status information from a device that does not contain an endpoint, such as a switch. The data payload is typically placed in a queue in the targeted endpoint, and an interrupt is typically generated to a local processor. A port-write request to a queue that is full or busy servicing another request may be discarded.

2.3.5 Doorbell Operation

The doorbell operation is shown in Figure 2-21. It consists of the DOORBELL and RESPONSE transactions (typically a DONE response), and is used by a processing element to send a very short message to another processing element through the interconnect fabric. The DOORBELL transaction contains the info field to hold information and does not have a data payload. This field is software-defined and can be used for any desired purpose; see the *RapidIO Interconnect Specification*, Section 3.1.4, *Type 10 Packet Formats (Doorbell Class)*, for information about the info field. A processing element that receives a doorbell transaction takes the packet and puts it in a doorbell message queue within the processing element. This queue may be implemented in hardware or in local memory. This behavior is similar to that of typical message passing mailbox hardware. The local processor is expected to read the queue to determine the sending processing element and the info field, and determine what action to take.

The DOORBELL functionality is user-defined, but this packet type is commonly used to initiate DSP core (CPU) interrupts. A DOORBELL packet is not associated with a particular data packet that was previously transferred, so the info field of the packet must be configured to reflect the DOORBELL bit to be serviced for the correct TID (transfer information descriptor) information to be processed.

Figure 2-21. Doorbell Operation



The DOORBELL packet's 16-bit INFO field indicates which DOORBELL register interrupt bit to set. There are four DOORBELL registers, each currently with 16 bits, allowing 64 interrupt sources or circular buffers. Each bit can be assigned to any core as described below by the Interrupt Condition Routing Registers. Additionally, each status bit is user-defined for the application. For instance, it may be desirable to support multiple priorities with multiple TID circular buffers per core if control data uses a high priority (for example, priority = 2), while data packets are sent on priority 0 or 1. This allows the control packets to have preference in the switch fabric and arrive as quickly as possible. Since it may be required to interrupt the CPU for both data and control packet processing separately, separate circular buffers are used, and DOORBELL packets need to distinguish between them for interrupt servicing. If any reserved bit in the DOORBELL info field is set, an error response is sent.

Figure 2-22. Examples of DOORBELL_INFO Designations

info Field Segments				Value Written To DOORBELL_INFO Field Of LSUn_REG5	Associated Doorbell Interrupt Routing Bits	Mapped To This Doorbell Interrupt Status Bit
Reserved	Doorbell Reg #	rsv	Doorbell Bit			
000000000b	00b	0b	0000b	0000h	DOORBELL0_ICRR[3:0]	DOORBELL0_ICSR[0]
000000000b	00b	0b	1001b	0009h	DOORBELL0_ICRR2[7:4]	DOORBELL0_ICSR[9]
000000000b	01b	0b	0111b	0027h	DOORBELL1_ICRR[31:28]	DOORBELL1_ICSR[7]
000000000b	01b	0b	1100b	002Ch	DOORBELL1_ICRR2[19:16]	DOORBELL1_ICSR[12]
000000000b	10b	0b	0101b	0045h	DOORBELL2_ICRR[23:20]	DOORBELL2_ICSR[5]
000000000b	10b	0b	1111b	004Fh	DOORBELL2_ICRR2[31:28]	DOORBELL2_ICSR[15]
000000000b	11b	0b	0110b	0066h	DOORBELL3_ICRR[27:24]	DOORBELL3_ICSR[6]
000000000b	11b	0b	1011b	006Bh	DOORBELL3_ICRR2[15:12]	DOORBELL3_ICSR[11]

2.3.6 Atomic Operations

The Atomic operation is a combination read and write operation. The destination reads the data at the specified address, returns the read data to the requestor, performs the required operation to the data, and then writes the modified data back to the specified address without allowing any intervening activity to that address. Defined operations are increment, decrement, test-and-swap, set, and clear. Of these, only test-and-swap requires the requesting processing element to supply data. Incoming Atomic operations which target the device are not supported for internal L2 memory or registers. Atomic request operations to external devices are supported and have a response packet.

Request Atomic operations (Ftype 2) never contain a data payload. These operations are similar to NREAD (24h) transactions. The data payload size for the response to an Atomic transaction is 8 bytes. The addressing scheme defined for the read portion of the Atomic transaction also controls the size of the atomic operation in memory, so that the bytes are contiguous and sized by byte, half-word (2 bytes), or word (4 bytes). They are aligned to that boundary and byte lane as with a regular read transaction. Double-word (8-byte), 3-byte, 5-byte, 6-byte, and 7-byte Atomic transactions are not allowed.

Atomic test-and-swap operations (Ftype 5) to external devices are limited to a payload of one double-word (8 bytes). These operations are like NWRITE with response (55h) transactions. The addressing scheme defined for the write transactions also controls the size of the Atomic operation in memory, so that the bytes are contiguous and sized by byte, half-word (2 bytes), or word (4 bytes). They are aligned to that boundary and byte lane as with a regular write transaction. Double-word (8-byte), 3-byte, 5-byte, 6-byte and 7-byte Atomic test-and-swap transactions are not allowed. Upon receipt of the request, the targeted device swaps the contents of the specified memory location and the payload if the contents of the memory location are all 0s. The contents of the memory location are returned, and the appropriate completion code is set in the LSU status register (LSU *n*_REG6).

2.3.7 Congestion Control

The RapidIO flow control specification is referenced in Table 2-22. This section describes the requirements and implementation of congestion control within the peripheral.

The peripheral is notified of switch fabric congestion through type 7 RapidIO packets. The packets are referred to as congestion control packets (CCPs). The purpose of these packets is to turn off (Xoff), or turn on (Xon) specific flows defined by DESTID and PRIORITY of outgoing packets. CCPs are sent at the highest priority in an attempt to address fabric congestion as quick as possible. CCPs do not have a response packet and they do not have guaranteed delivery.

When the peripheral receives an Xoff CCP, the peripheral must block both outgoing LSU and CPPI packets destined for that flow. When the peripheral receives an Xon, the flow may be enabled. Since CCPs may arrive from different switches within the fabric, it is possible to receive multiple Xoff CCPs for the same flow. For this reason, the peripheral must maintain a table and count of Xoff CCPs for each flow. For example, if two Xoff CCPs are received for a given flow, two Xon CCPs must be received before the flow is enabled.

Because CCPs do not have guaranteed delivery and can be dropped by the fabric, an implicit method of enabling an Xoff'd flow must exist. A simple timeout method is used. Additionally, flow control checks can be enabled or disabled through the transmit source flow control masks. Received CCPs do not get passed through the VBUS interface.

2.3.7.1 Detailed Description

To avoid large and complex table management, a basic scheme is used for implementation of the RIO congestion management. The primary goal is to avoid large parallel searches of a centralized congested route table for each out-going packet request. The congested route table requirements and subsequent searches would be very overwhelming if all possible DESTID and PRIORITY combinations each had its own entry. To implement a more basic scheme, the following assumptions have been made:

- A small number of flows constitute the majority of traffic, and these flows are most likely to be causing congestion
- HOL blocking is undesired, but allowable for TX CPPI queues
- Flow control is based on DESTID only, regardless of PRIORITY

The congested route table is therefore more static in nature. Instead of dynamically updating a table with each CCP's flow information as it arrives, a small finite-entry table is set up and configured by software to reflect the more critical flows it is using. Only these flows have a discrete table entry. A 16-entry table is used to reflect 15 critical flows, leaving the sixteenth entry for general "other flows", which are lumped together. Figure 2-23 shows the MMR table entries that are programmable by the CPU through the VBUSP configuration bus. A 3-bit hardware counter is implemented for table entries 0 through 14, to maintain a count of Xoff CCPs for that flow. The "other flows" table entry counts Xoff CCPs for all flows other than the discrete entries. The counter for this table entry is 5-bit. All out-going flows with non-zero Xoff counts, are disabled. The counter value is decremented for each corresponding Xon CCP that is received, but it should not decrement below zero. Additionally, a hardware timer is needed for each table entry to turn on flows that may have been abandoned by lost Xon CCPs. The timer value must be an order of magnitude larger than the 32b port response timeout CSR value. For this reason, each transmission source adds two bits to its 4-bit response time-out counter described in Section 2.3.3.2 and Section 2.3.3.1. The additional two bits are used to count three timecode revolutions and provide an implicit Xon timer equal to 3x the response time-out counter value.

Figure 2-23. Flow Control Table Entry Registers (Address offset 0x07D0 — 0x080C)

Register Name	31-18	17-16	15-0
RIO_FLOW_CNTL0	Reserved	tt	Flow_Cntl_ID0
RIO_FLOW_CNTL1	Reserved	tt	Flow_Cntl_ID1
. . . .			
RIO_FLOW_CNTL15	Reserved	tt	Flow_Cntl_ID15
	R, All zeros	R/W, 0b01	R/W. 0x0000

Table 2-22. Flow Control Table Entry Registers (Address offset 0x07D0 — 0x080C)

Name	Bit	Access	Reset Source	Reset Value	Description
tt	[17-16]	R/W	VBUS_rst	0b01	Selects Flow_Cntl_ID length. • 0b00 — 8b IDs • 0b01 — 16b IDs • 0b10 — 0b11 — reserved
Flow_Cntl_ID0	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 0
Flow_Cntl_ID1	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 1
Flow_Cntl_ID2	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 2
Flow_Cntl_ID3	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 3
Flow_Cntl_ID4	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 4
Flow_Cntl_ID5	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 5
Flow_Cntl_ID6	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 6
Flow_Cntl_ID7	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 7
Flow_Cntl_ID8	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 8
Flow_Cntl_ID9	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 9
Flow_Cntl_ID10	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 10
Flow_Cntl_ID11	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 11
Flow_Cntl_ID12	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 12
Flow_Cntl_ID13	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 13
Flow_Cntl_ID14	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 14
Flow_Cntl_ID15	[15-0]	R/W	VBUS_rst	0x0000	DestID of Flow 15, if all 0's this table entry represents all flows other than flows 0 - 14

Each transmit source, including LSU, or Tx CPPI queue, indicates which of the 16 flows it uses by having a mask register. Table 2-23 lists the required 16-bit registers with bit masks. The CPU must configure these registers upon reset. The default setting is all 1s, indicating that the transmit source supports all flows. If the register is set to all 0s, the transmit source is claiming that it does not support any flow, and consequently that source is never flow-controlled. If any of the table entry counters that a transmit source

supports, have a corresponding non-zero Xoff count, the transmit source is flow-controlled. A simple 16-bit bus indicates the Xoff state of all 16 flows and is compared to the transmit source mask register. Each source interprets this result and performs flow control accordingly. For example, a LSU module that is flow-controlled can reload its registers and attempt to send a packet to another flow, while a TX CPPI queue that is flow-controlled may create HOL-blocking issues on that queue. The following registers are reset when vbus_rst is asserted.

Table 2-23. Transmit Source Flow Control Masks

Name of Register	Reset Value	Bits[31-16]	Bits[15-0]
RIO_LSU_FLOW_MASKS0	32'hFFFFFFFF	LSU1 Flow Mask	LSU0 Flow Mask
RIO_LSU_FLOW_MASKS1	32'hFFFFFFFF	LSU3 Flow Mask	LSU2 Flow Mask
RIO_LSU_FLOW_MASKS2	32'hFFFFFFFF	LSU5 Flow Mask	LSU4 Flow Mask
RIO_LSU_FLOW_MASKS3	32'hFFFFFFFF	LSU7 Flow Mask	LSU6 Flow Mask
RIO_TX_CPPI_FLOW_MASKS0	32'hFFFFFFFF	TX Queue1 Flow Mask	TX Queue0 Flow Mask
RIO_TX_CPPI_FLOW_MASKS1	32'hFFFFFFFF	TX Queue3 Flow Mask	TX Queue2 Flow Mask
RIO_TX_CPPI_FLOW_MASKS2	32'hFFFFFFFF	TX Queue5 Flow Mask	TX Queue4 Flow Mask
RIO_TX_CPPI_FLOW_MASKS3	32'hFFFFFFFF	TX Queue7 Flow Mask	TX Queue6 Flow Mask
RIO_TX_CPPI_FLOW_MASKS4	32'hFFFFFFFF	TX Queue9 Flow Mask	TX Queue8 Flow Mask
RIO_TX_CPPI_FLOW_MASKS5	32'hFFFFFFFF	TX Queue11 Flow Mask	TX Queue10 Flow Mask
RIO_TX_CPPI_FLOW_MASKS6	32'hFFFFFFFF	TX Queue13 Flow Mask	TX Queue12 Flow Mask
RIO_TX_CPPI_FLOW_MASKS7	32'hFFFFFFFF	TX Queue15 Flow Mask	TX Queue14 Flow Mask

Table 2-24. Transmit Source Flow Control Masks

Name	Bit	Access	Description
Flow Mask	0	R/W	0b0 — TX source doesn't support Flow0 from table entry 0b1 — TX source does support Flow0 from table entry
Flow Mask	1	R/W	0b0 — TX source doesn't support Flow1 from table entry 0b1 — TX source does support Flow1 from table entry
Flow Mask	2	R/W	0b0 — TX source doesn't support Flow2 from table entry 0b1 — TX source does support Flow2 from table entry
Flow Mask	3	R/W	0b0 — TX source doesn't support Flow3 from table entry 0b1 — TX source does support Flow3 from table entry
Flow Mask	4	R/W	0b0 — TX source doesn't support Flow4 from table entry 0b1 — TX source does support Flow4 from table entry
Flow Mask	5	R/W	0b0 — TX source doesn't support Flow5 from table entry 0b1 — TX source does support Flow5 from table entry
Flow Mask	6	R/W	0b0 — TX source doesn't support Flow6 from table entry 0b1 — TX source does support Flow6 from table entry
Flow Mask	7	R/W	0b0 — TX source doesn't support Flow7 from table entry 0b1 — TX source does support Flow7 from table entry
Flow Mask	8	R/W	0b0 — TX source doesn't support Flow8 from table entry 0b1 — TX source does support Flow8 from table entry
Flow Mask	9	R/W	0b0 — TX source doesn't support Flow9 from table entry 0b1 — TX source does support Flow9 from table entry
Flow Mask	10	R/W	0b0 — TX source doesn't support Flow10 from table entry 0b1 — TX source does support Flow10 from table entry
Flow Mask	11	R/W	0b0 — TX source doesn't support Flow11 from table entry 0b1 — TX source does support Flow11 from table entry
Flow Mask	12	R/W	0b0 — TX source doesn't support Flow12 from table entry 0b1 — TX source does support Flow12 from table entry
Flow Mask	13	R/W	0b0 — TX source doesn't support Flow13 from table entry 0b1 — TX source does support Flow13 from table entry

Table 2-24. Transmit Source Flow Control Masks (continued)

Name	Bit	Access	Description
Flow Mask	14	R/W	0b0 — TX source doesn't support Flow14 from table entry 0b1 — TX source does support Flow14 from table entry
Flow Mask	15	R/W	0b0 — TX source doesn't support Flow15 from table entry 0b1 — TX source does support Flow15 from table entry

The information in Table 2-24 for bits [15-0] applies similarly to bits [31-16] of the flow control mask register.

2.3.8 Endianness

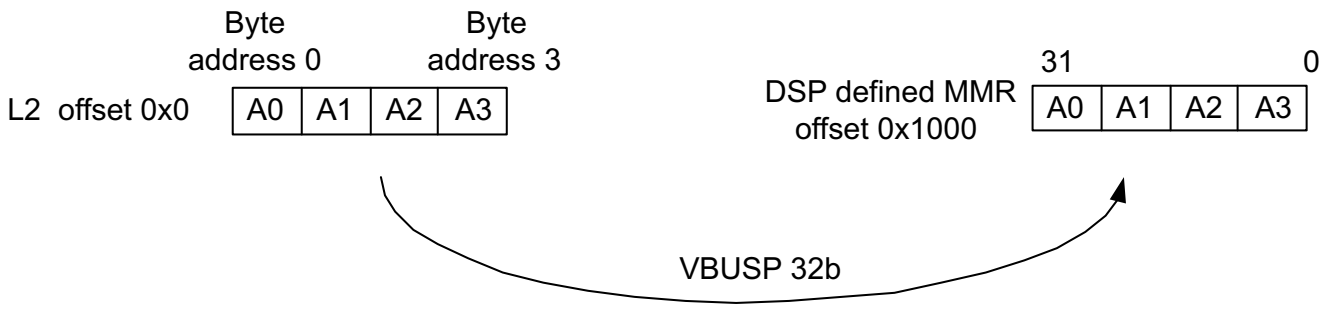
RapidIO is based on big-endian. This is discussed in detail in Section 2.4 of the *RapidIO Interconnect Specification*. Essentially, big-endian specifies the address ordering as the most significant bit/byte first. For example, in the 29-bit address field of a RapidIO packet, the left-most bit that is transmitted first in the serial bit stream is the MSB of the address. Likewise, the data payload of the packet is double-word-aligned big-endian, which means the MSB is transmitted first. Bit 0 of all the RapidIO-defined MMR registers is the MSB.

All endian-specific conversion is handled within the peripheral. For double-word-aligned payloads, the data should be written contiguously into memory beginning at the specified address. Any unaligned payloads are padded and properly aligned within the 8-byte boundary. In this case, WDPTR, RDSIZE, and WRSIZE RapidIO header fields indicate the byte position of the data within the double-word boundary. An example of an unaligned transfer is shown in Section 2.4 of the *RapidIO Interconnect Specification*.

2.3.8.1 Translation for Memory-Mapped Register Space

There are no endian translation requirements for accessing the local memory mapped register space. Regardless of the device memory endian configuration, all configuration bus accesses are performed on 32-bit values at a fixed address position. The bit positions in the 32-bit word are defined by this specification. This means that a memory image copied to a memory-mapped register is identical between little-endian and big-endian configurations. Configuration bus reads are performed in the same manner. Figure 2-24 shows the concept. The desired operation is to locally update a serial RapidIO MMR (offset 1000h) with a value of A0A1A2A3h, using the configuration bus.

Figure 2-24. Configuration Bus Example

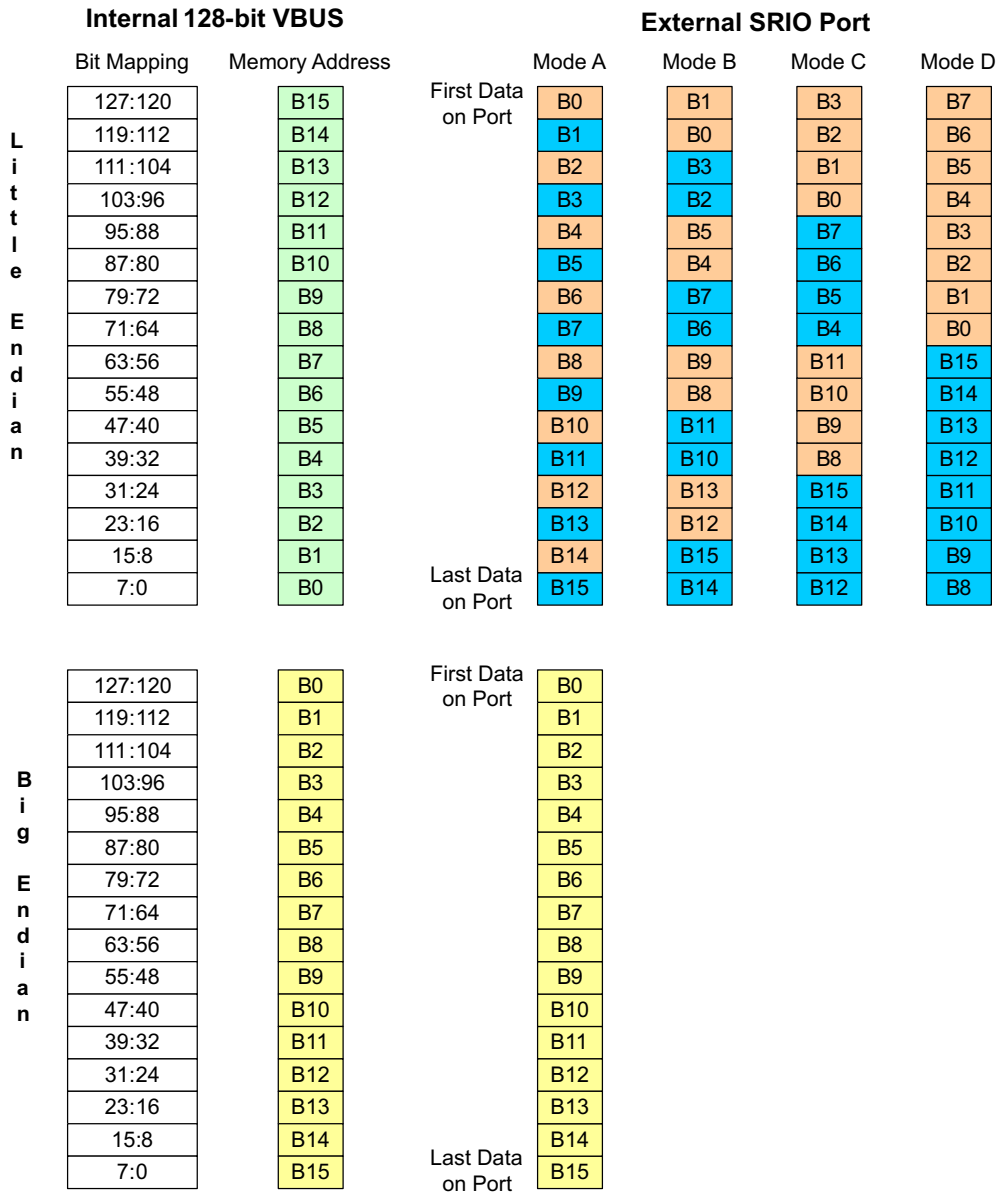


When accessing a RapidIO-defined, memory-mapped register within an external device, RapidIO allows 4 bytes, 8 bytes, or any multiple of a double-word access (up to 64 bytes) for type 8 (maintenance) packets. The peripheral only supports 4-byte accesses as the target, but can generate all sizes of request packets. RapidIO is defined as big-endian only, and has double-word-aligned big-endian packet payloads.

2.3.8.2 Translation for Payload Data

The DMA also supports byte-wide accesses. The peripheral performs endian conversion on the payload if little-endian is used on the device. This conversion is not only applicable for type 8 packets, but is relevant for all outgoing payloads of NWRITE, NWRITE_R, SWRITE, NREAD, and message passing. Based on the application model, when run in little-endian mode, swapping the original big-endian data based on different boundaries can be done: swap on 8-byte boundary, swap on 4-byte boundary, swap on 2-byte boundary, and swap on one-byte boundary as shown in Figure 2-25.

Figure 2-25. Translation for Payload Data



The little-endian data swapping mode for LSU, MAU, and message passing (TXU/RSU) can all be set differently in the PER_SET_CNTL register.

2.3.9 Interrupt Operation

This section describes the interrupt capabilities of the peripheral.

2.3.9.1 General Description

The SRIO block has 25 output interrupt lines. They are grouped and named as follows:

- INTDST0 – INTDST15: General purpose interrupt lines
- INTDST16 – INTDST23: Dedicated doorbell interrupt lines (No interrupt pacing)
- RapidIO_INT_CDMA_0: PKTDMA starvation interrupt line

Figure 2-26. Interrupt Mapping to CPU

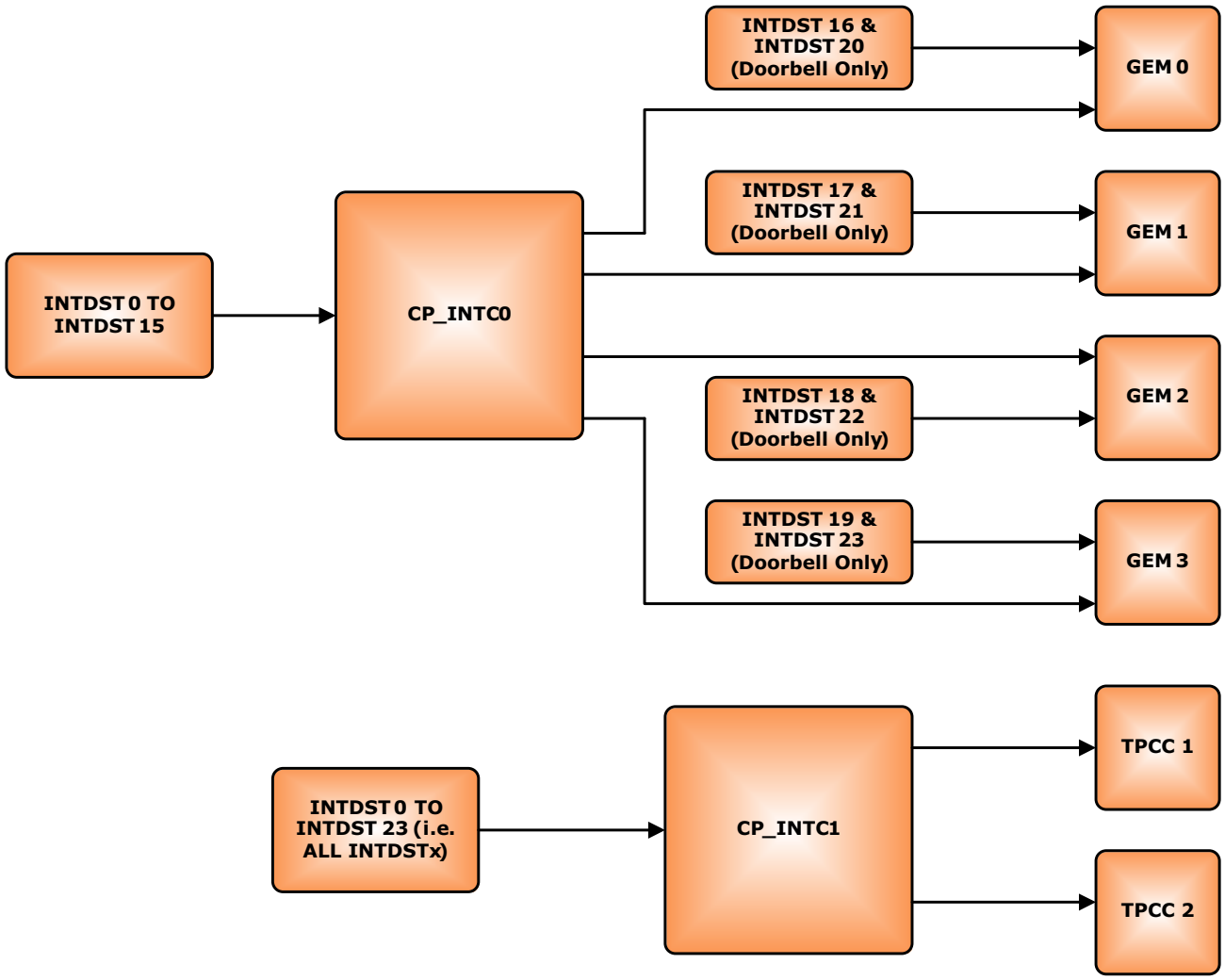


Figure 2-26 shows the interrupt mapping to CPU and EDMA. The interrupt conditions that can be configured to generate an interrupt fall into the following categories.

- CPU servicing: Event indicating that the CPU should service the peripheral.
- Error Status: Event indicating that a run-time error has occurred. The CPU should reset and resynchronize the peripheral.
- Critical Error: Event indicating that a critical error has occurred. The CPU should reset the system.

2.3.9.1.1 Direct I/O (Doorbell) Servicing Interrupts

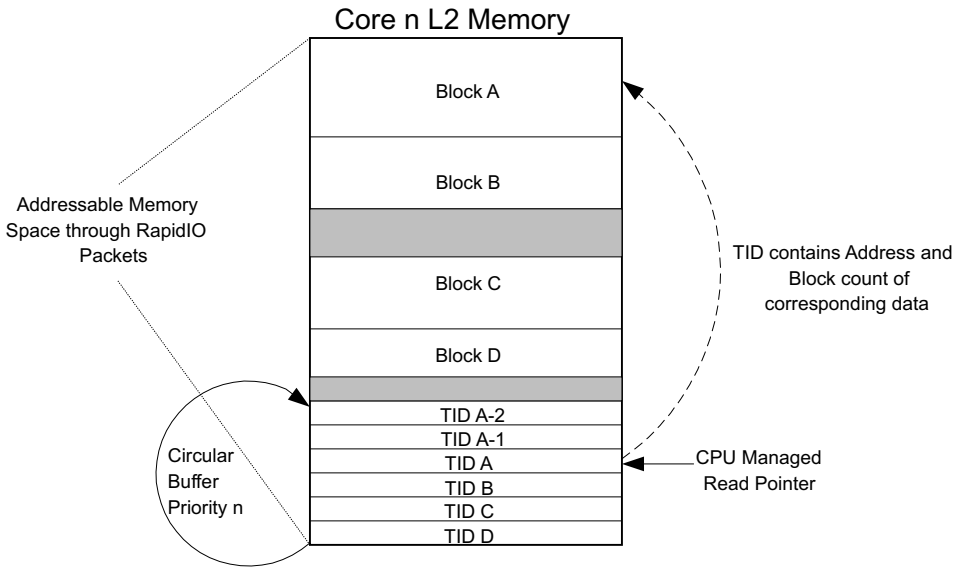
Because RapidIO is a packet-oriented interface, the peripheral must recognize and respond to in-band interrupt signals from the serial interface. There are no GPIO or external pins used to indicate an interrupt request.

CPU servicing interrupts lag behind their corresponding data, which was generally transferred from an external processing element into local L2 memory. Using the direct I/O protocol, once the single or multipacket data transfer is complete, the external PE, or the peripheral itself, must notify the local processor that the data is available for processing. To avoid erroneous data from being processed by the local CPU, it is extremely important to ensure the completion of the data transfer through the DMA before the CPU interrupt is serviced. This condition could easily occur because the data and interrupt queues are independent of each other, and DMA transfers can stall. To avoid this condition, all data transfers from the peripheral through the DMA use write-with-response VBUSM commands. This allows the peripheral to always know that outstanding transfers have completed. Interrupts are generated only after all VBUSM responses are received. Because all RapidIO packets are handled sequentially and submitted on the same DMA priority queue, the requirement imposed is that the peripheral keep track of the number of TR requests submitted and the number of responses received. In this manner, a simple counter within the peripheral can be used to ensure that data packets have arrived in memory before submitting an interrupt.

Another important aspect of CPU servicing interrupts is addressing the correct core in multicore devices. For example, one goal of a multicore device is to appear as a single core device to all external components. The requirement is that an interrupt can be generated from any 1X port to any of the internal cores. This requirement is satisfied by using a DOORBELL in the direct I/O mode.

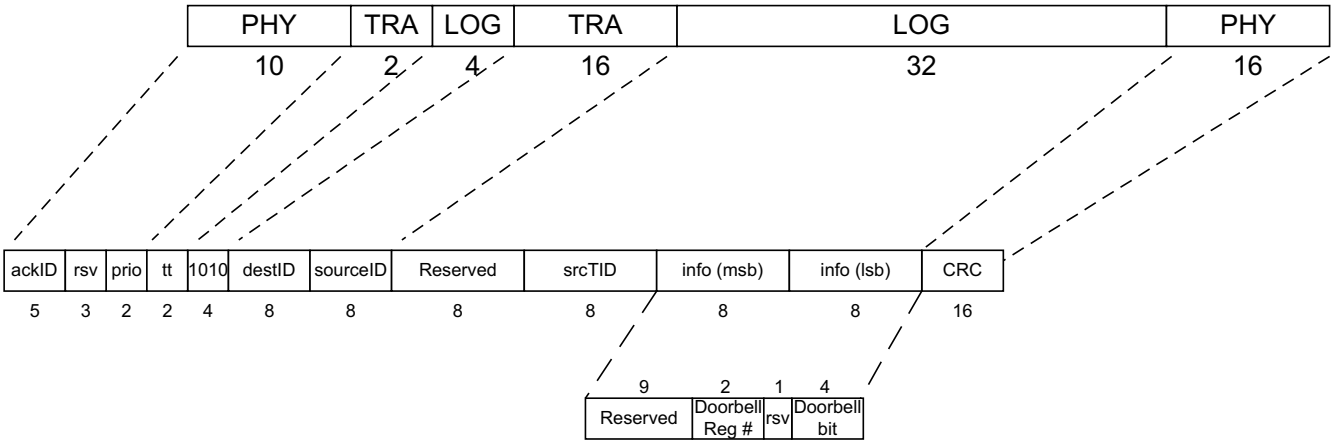
When the direct I/O mode is used for data transfer, there is already knowledge of which core should process the interrupt, based on the data's L2 address range. For example, in Figure 2-27 there are two transfers—the raw data, and transfer information descriptor (TID) info. TID contains information about the raw data including block count, address, and implementation specific details. The TID info is written into a circular buffer within L2, with read pointers which are managed by the CPU software. When an interrupt is issued to a core, as described above, the core reads the TID corresponding to the current read pointer and then processes the corresponding data. Multiple circular buffers can be tracked; for example, a circular buffer per input port would be needed to avoid ports over-writing each other's TID info and to maintain the correct CPU read pointers.

Figure 2-27. Direct I/O Circular Buffer Scheme for Interrupt Management



The sending device initiates the interrupt by using the RapidIO-defined DOORBELL message. The DOORBELL packet format is shown in Figure 2-28. The DOORBELL functionality is user-defined, but it is common practice in the industry to use this packet type to initiate CPU interrupts. A DOORBELL packet is not associated with a particular data packet that was previously transferred, thus the INFO field of the packet must be configured to reflect the DOORBELL bit to be serviced, so that the correct TID info is processed.

Figure 2-28. RapidIO DOORBELL Packet For Interrupt Use



The DOORBELL packet's 16-bit INFO field is used to indicate which DOORBELL register and which interrupt bit in that register to set. There are four DOORBELL registers, each with 16 bits, allowing 64 interrupt sources (or circular buffers). Each bit can be assigned to one of the output interrupt lines through the proper Interrupt Condition Routing register. The Interrupt Control register determines whether the doorbell interrupts are routed to the general purpose interrupt lines or the dedicated doorbell interrupt lines.

Additionally, each status bit is user-defined for the application. For instance, it may be desirable to support multiple priorities with multiple TID circular buffers per core if control data uses a high priority (such as priority = 2), while data packets are sent on priority 0 or 1. This allows the control packets to have preference in the switch fabric and arrive as quickly as possible. Because it may be required to interrupt the CPU for both data and control packet processing separately, separate circular buffers are used, and DOORBELL packets must distinguish between them for interrupt servicing. If any reserved bit in the DOORBELL info field is set, an error response is sent. Interrupt pacing for direct I/O is implemented by the SRIO peripheral to manage the interrupt rate.

2.3.9.1.2 Message Passing Servicing Interrupts

The interrupt approach in the message-passing protocol is somewhat different. Because the source device has no idea where the data is physically stored in the destination device, and as each messaging packet contains size and segment information, the interrupt can be automatically generated after all packet segments comprising the complete message have been successfully received. The communications port programming interface (CPPI) DMA is used to transfer the data to its destination. Essentially, this is a link-list approach versus a circular buffer approach. Data buffer descriptors which contain information such as start of packet (SOP), end of packet (EOP), and packet length are built from the RapidIO header fields. The data buffer descriptors also contain the address of the corresponding data buffer as assigned by the receive device. The data buffer descriptors are then link-listed together as multiple packets are received and placed in a CPPI queue through the queue manager. Interrupts are generated by the queue manager after all segments of a message (or multiple messages) are received. Interrupt pacing for message passing is also implemented by the queue manager sub-system to manage the interrupt rate.

2.3.9.1.3 Error Interrupts

Error handling on the RapidIO link is handled by the peripheral, and as such, does not require the intervention of software for recovery. This includes CRC errors due to bit-rate errors that may cause erroneous or invalid operations. The exception to this statement is the use of the RapidIO error management extended features. This specification monitors and tabulates the errors that occur on a per port basis. If the number of errors exceeds a pre-determined configurable amount, the peripheral should interrupt the CPU software and notify that an error condition exits. Alternatively, if a system host is used, the peripheral may issue a port-write operation to notify the system software of a bad link.

A system reset or Critical Error interrupt can be initialized through the RapidIO link. This procedure allows an external device to reset the local device, causing all state machine and configuration registers to reset to their original values. This is executed with the Reset-Device command described in Part VI, Section 3.4.5 of the Serial specification. Four sequential Reset-Device control symbols are needed to avoid inadvertent resetting of a device.

2.3.9.2 Interrupt Registers

This section presents a summary of the various interrupt registers. Detailed descriptions can be found in *Chapter 3*.

There is one Interrupt Status Decode register (ISDRn) for each of the INTDSTn output interrupt lines. These registers show the interrupt group (and specific bit in the case of doorbell interrupts) and minimize the total number of register reads required to determine the interrupt source.

All peripheral conditions that can result in a CPU interrupt are grouped so that the interrupt can be accessed in the minimum number of register reads possible. The interrupt groups are doorbell, LSU, and error. Each of these groups has one or multiple sets of the following registers:

- Interrupt Condition Routing Register (ICRR): Command register that determines which outputs interrupt line is assigned to each interrupt condition.
- Interrupt Condition Status Register (ICSR): Status register that reflects the state of each condition that can trigger the interrupt. Also writeable for test purposes.
- Interrupt Condition Clear Register (ICCR): Command register that allows each condition to be cleared. This is typically required prior to enabling a condition, such that spurious interrupts are not generated.

The Interrupt Control register (INTERRUPT_CTL) determines whether the doorbell interrupts are routed to the general purpose interrupt lines or the dedicated doorbell interrupt lines.

The Interrupt Rate Control registers (INTDSTn_RATE_CNTL and INTDST_RATE_DIS) are used to control the pace of interrupt generation on the INTDST0 – INTDST15 interrupt lines. This feature is not available on INTDST16 – INTDST23.

2.3.9.3 Interrupt Handling

When the CPU is interrupted, it reads the ISDRn and ICSR registers to determine the source of the interrupt and appropriate action to take. For example, if it is a DOORBELL interrupt, the CPU reads from an L2 address specified by its circular buffer read pointer managed by software. There may be more than one circular buffer for each core. The correct circular buffer to read from and increment depends on the bit set in the ICSR register. The CPU then clears the status bit.

For Error Status interrupts, the peripheral must indicate to all the CPUs that one of the link ports has reached the error threshold. In this case, the peripheral sets the status bit indicating degraded or failed limits have been reached, and an interrupt is generated to each core through the ICRR mapping. The cores can then scan the ICSR registers to determine the port with the error problems. Further action can then be taken as determined by the application.

2.3.9.4 Interrupt Pacing

The rate at which an interrupt can be generated is controllable for each physical interrupt destination. Rate control is implemented with a programmable down-counter. The load value of the counter is written by the CPU into the registers shown in RIO_INTDSTn_RATE_CNTL register. The counter re-loads and immediately starts down-counting each time the CPU writes these registers. Once the rate control counter register is written, and the counter value reaches zero (the CPU may write zero immediately for a zero

count), the interrupt pulse generation logic is allowed to fire a single pulse if any bits in the corresponding ICSR register bits are set (or become set after the zero count is reached). The counter remains at zero. Once the single pulse is generated, the logic does not generate another pulse, regardless of interrupt status changes, until the rate control counter register is written again. If interrupt pacing is not desired for a particular INTDST, it can be disabled using RIO_INTDST_RATE_DIS. If an ICSR is not mapped to an interrupt destination, pending interrupt bits within the ICSR maintain current status. Once enabled, the interrupt logic re-evaluates all pending interrupts and re-pulses the interrupt signal if any interrupt conditions are pending. The down counter is based on the DMA clock cycle.

2.3.10 Reset and Powerdown

The RapidIO peripheral allows independent software controlled shutdown for the logical blocks listed in Table 2-25. With the exception of BLK0_EN for the memory-mapped registers (MMRs), when the BLK *n*_EN signals are de-asserted, the clocks are gated to these blocks, effectively providing a shutdown function.

Table 2-25. Reset Hierarchy

Logical Block	Bus Reset	Non-POR Reset	GB _EN	BLK0 _EN	BLK 1 _EN	BLK2 _EN	BLK3 _EN	BLK4 _EN	BLK5 _EN	BLK6 _EN	BLK7 _EN	BLK8 _EN
DMA VBUSM i/f (CAU)	X	X	X									
MMR — Reset/PD Ctl Registers	X											
MMR — non-Reset/PD Ctl Registers (Logical Block 0)	X	X	X	X								
Interrupt Handling Unit (IHU)	X	X	X									
Traffic Flow Logic	X	X	X									
LSU (Direct I/O Initiator)	X	X	X		X							
MAU (Direct I/O Target)	X		X			X						
TXU (Message Passing Initiator)	X	X	X				X					
RXU (Message Passing Target)	X	X	X					X				
Congestion Control Unit (CCU)	X	X	X									
Port 0 Datapath	X		X						X			
Port 1 Datapath	X		X							X		
Port 2 Datapath	X		X								X	
Port 3 Datapath	X		X									X

Reset of the SerDes macros is handled independently of the registers discussed in this section. The SerDes can be configured to shutdown unused links or fully shutdown. SerDes TX and RX channels may be enabled/disabled by writing to bit 1/0 of the ENTX and ENRX bit fields of the SerDes registers respectively. The PLL and remaining SerDes functional blocks can be controlled by writing to the ENPLL signal in the SerDes configuration control register. This bit will drive the SerDes signal input, which will gate the reference clock to these blocks internally. This reference clock is sourced from a device pin specifically for the SerDes and is not derived from the CPU clock, thus it resets asynchronously. ENPLL will disable all SerDes high-speed output clocks. Since these clocks are distributed to all the links, ENPLL should only be used to completely shutdown the peripheral. It should be noted that shutdown of SerDes links in between normal packet transmissions is not permissible for two reasons. First, the serial RapidIO sends idle packets between data packets to maintain synchronization and lane alignment. Without this mechanism, the RapidIO RX logic can be mis-aligned for both 1X and 4X ports. Second, the lock time of the SerDes PLL would need to reoccur, which would slow down the operation.

When the SerDes ENTX signal is held low, the corresponding transmitter is powered down. In this state, both outputs, TXP and TXN, will be pulled high to VDDT.

2.3.10.1 Reset and Powerdown Summary

After reset, the state of the peripheral depends on the default register values.

Software can also perform a hard reset of each logical block within the peripheral through the GBL_EN and BLK *n*_EN bits. The GBL_EN bit resets the peripheral, while the rest of the device is not reset. The BLK *n*_EN bits shut down unused portions of the peripheral, which minimizes power by resetting the appropriate logical blocks and gating off the clock to the appropriate logical blocks. This should be considered an abrupt reset independent of the state of the peripheral and that resets the peripheral to its original state.

Upon reset of the peripheral, the device must reestablish communication with its link partner. Depending on the system, this may include a discovery phase in which a host processor reads the peripheral's CAR/CSR registers to determine its capabilities. In its simplest form, it involves retraining the SerDes and going through the initialization phase to synchronize on bit and word boundaries by using idle and control symbols, as described in Section 5.5.2 of the Part VI of the *RapidIO Interconnect Specification*. Until the peripheral and its partner are fully initialized and ready for normal operation, the peripheral will not send any data packets or non-status control symbols.

- GBL_EN: Resets all MMRs, excluding Reset Ctl Values (0000h-017Fh). Resets all logical blocks except MMR configuration bus i/f. While asserted, the slave configuration bus is operational.
- Non-POR Reset: Resets all MMRs, excluding Reset Ctl Values (0000h-017Fh). Resets all logical blocks except MAU and MMR configuration bus i/f. While asserted, the slave configuration bus is operational.
- BLK_EN0: Resets all MMRs, excluding Reset Ctl Values (0000h-017Fh). Other logical blocks are unaffected, including MMR configuration bus i/f.
- BLK_EN[n:1]: Single enable/reset per logical block.

The following list describes the logical blocks in the SRIO module:

- BLOCK 0 - MMRs
- BLOCK 1 - LSU
- BLOCK 2 - MAU
- BLOCK 3 - TXU
- BLOCK 4 - RXU
- BLOCK 5 - Port 0
- BLOCK 6 - Port 1
- BLOCK 7 - Port 2
- BLOCK 8 - Port 3

See *Chapter 3* for the definition of the Enable and Enable Status registers.

2.3.10.2 Software Shutdown Details

Power consumption is minimized for all logical blocks in shutdown. In addition to simply asserting the appropriate reset signal to each logical block within the peripheral, clocks are also gated off to the corresponding logical block. Clocks are allowed to run for 32 clock cycles, which is necessary to fully reset each logical block. When the appropriate logical block is fully reset, the clock input to that sub-block is gated off. When software asserts GBL_EN/BLKn_EN to release the logical block from reset, the clocks are un-gated and the GBL_EN_STAT/BLKn_EN_STAT bits indicate a value of 1b.

NOTE: The BLK_EN bits allow the user to shut down and gate clocks to unused portions of the logic, while other parts of the peripheral continue to operate. When shutting down an individual block, if TXU and RXU queues are not torn down correctly, the DMA bus could hang. For example, setting BLK3_EN = 0 (disabling the TXU) before a teardown of the queue could cause any outstanding DMA request returned to the peripheral for the TXU to hang the bus.

When using the GBL_EN to shutdown or reset the entire peripheral, it is important to first stop all master-initiated commands on the DMA bus interface. For example, if the GBL_EN is asserted in the middle of a DMA transfer from the peripheral, this could hang the bus. The procedure to follow is:

1. Stop all RapidIO source transactions, including LSU and TXU operations. The four LSU blocks should indicate a BSY status of 0b. If an EDMA channel is used for driving the LSU, it must be stopped to prevent new or additional transfers. This procedure is outside the scope of this specification. Teardown of the TXU queues is accomplished by writing to the appropriate registers in the PKTDMA. Hardware then tears down the queues and clears these bits automatically when complete.
2. Stop all RapidIO message receive (RXU) operations. Teardown of the RXU queues is accomplished by writing to the appropriate registers in the PKTDMA. Hardware then tears down the queues and clears these bits automatically when complete.
3. Once teardown is complete, disable the PEREN bit of the PCR register to stop all new logical layer transactions.
4. Wait one second to finish any current DMA transfer.
5. De-assert GBL_EN

2.3.11 Reset Isolation for RapidIO

At times it will be necessary to reset a device that is no longer functioning correctly. In some of these cases, the need for reset requires that the memory image be reloaded or that logic within the device be taken to a state that can only be achieved with device reset. In those cases, it is desired to continue the RapidIO operation at a level that does not affect the remaining devices on the same RapidIO daisy chain. Whatever the cause or purpose, the DMA (VBUS) clock must remain stable during reset and RapidIO must continue operation as normal within the guidelines described below.

There are two essential requirements at the device level. One is a distinction between an overall device reset that includes RapidIO, and a device reset that does not include RapidIO. The second is a stable VBUS clock. Thus, there are two types of resets:

- Device reset with continued RapidIO operation
- RapidIO reset with continued device operation which is done by software

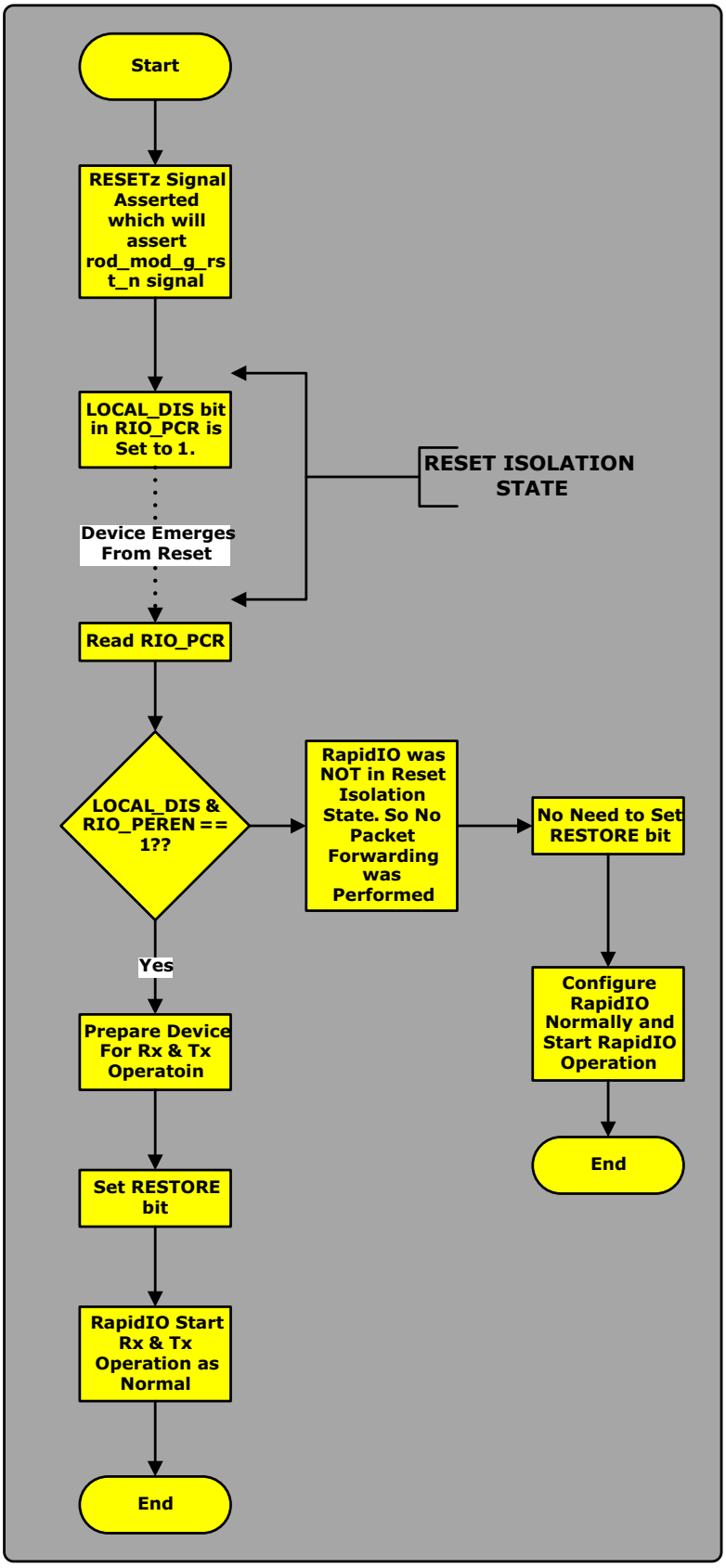
There are two device reset signals with which “Device Reset With Continued RapidIO Operation” can be handled:

- Reset of entire device with RapidIO (PORz)
- Reset of entire device excluding RapidIO along with other Reset Isolation peripherals (RESETz)

2.3.11.1 Device Reset with Continued RapidIO Operation

Two additional bits, one status bit and one command bit, are required to indicate the local disabled status of RapidIO and to restore normal operation. The assertion of RESETz is captured and stored as a status bit (LOCAL_DIS) in an MMR. This status bit, when set to 1, indicates that packets normally stored in memories associated with this device are to be discarded while packet forwarding remains active. When the device has emerged from reset, reads of this MMR and RIO_PCR should be performed to determine the state of LOCAL_DIS and PEREN. If LOCAL_DIS is 1 and PEREN is also 1, then RapidIO is active but packets for this device are being discarded. To restore reception of packets for this device, the RESTORE command bit in an MMR should be written as a 1. When the necessary state changes have been made to allow storage of packets to memories associated with this device, the final effect of the operation initiated by RESTORE should be to clear both the LOCAL_DIS status bit and the RESTORE bit. Figure 2-29 shows the complete operation in detail:

Figure 2-29. Reset Isolation



During Reset Isolation state following steps occur in sequence:

- All incomplete VBUS transactions are aborted.
- LOCAL_DIS is set to 1
- Transmit packets in the shared or FIFO buffers are sent normally.
- Discard all receive packets in shared or FIFO buffers whose DestID does not match one of the forwarding DestIDs.
- Clear the RIO_BLK_en bits for the LSU, RXU, and TXU. Only MAU is enable

2.3.11.2 RapidIO Reset with Continued Device Operation

It may be necessary to reset the RapidIO interface network without aborting the processing on devices attached to that network. This is intended to enable the physical RapidIO network to reestablish connections.

An available external interrupt source should be chosen to represent the need to reset RapidIO. Because the actual management of the reset operation is the responsibility of software, alternative means other than an external interrupt source may also be used. Proper software shutdown procedure must be used to perform the reset.

2.3.12 RX Multicast and Multiple DestID Support

The RapidIO peripheral supports RX multicast and unicast operations. To support multicast, an endpoint must either be able to accept discrete multiple DestIDs from incoming packets, or accept all DestIDs without performing a check on the incoming packet. In the latter case, the system relies on the fabric to deliver the appropriate packets to the endpoint (correct switch routing tables). The RapidIO peripheral supports both approaches by utilizing three configuration bits (described below) to control DestID checking. Table 2-26 lists the various configurations. The device's main baseID (0x1060) is now automatically copied into the RIO_DEVICEID_REG1 by hardware, and does not require a separate write by software.

The RIO_DEVICEID_REGn (n = 1 to 15) are inputs to the logical layer and not part of the MMR space. They inherit the values found in the Base Routing registers.

Table 2-26. Dest ID Checking for Multicast Operations

Mode	Operation	log_tgt_id_dis (PER_SET_CNTL, Bit 27, 0x0014)	tgt_id_dis (TLM_SP{0..3}_CONTROL Bit 21, 0x1b380, 0x1b400, 0x1b480, 0x1b500)	mtc_tgt_id_dis (TLM_SP{0..3}_CONTROL Bit 20, 0x1b380, 0x1b400, 0x1b480, 0x1b500)
A	Host can perform maintenance enumeration, discovery, and assignment using only DESTID = RIO_BASE_ID (0x1060) or one of the other 15 device ID value. No multicast No packet forwarding	X	0	0
B	Host can perform maintenance enumeration, discovery, and assignment using any DESTID. Allows hard coding (pin strapping) of BASE_ID, and then maintenance read back by the host. No multicast No packet forwarding	X	0	1
C	Host can perform maintenance enumeration, discovery, and assignment using only DESTID = RIO_BASE_ID (0x1060) or one of the other 15 device ID value. Supports 8 local multicast groups Supports packet forwarding for all packet types	0	1	0

Table 2-26. Dest ID Checking for Multicast Operations (continued)

Mode	Operation	log_tgt_id_dis (PER_SET_CNTL, Bit 27, 0x0014)	tgt_id_dis (TLM_SP{0..3}_CONTROL Bit 21, 0x1b380, 0x1b400, 0x1b480, 0x1b500)	mtc_tgt_id_dis (TLM_SP{0..3}_CONTROL Bit 20, 0x1b380, 0x1b400, 0x1b480, 0x1b500)
D	Host can perform maintenance enumeration, discovery, and assignment using any DESTID. Allows hard coding (pin strapping) of BASE_ID, and then maintenance read back by the host. Supports 8 local multicast groups No packet forwarding of Ftype 8 maintenance packets	0	1	1
E	Host can perform maintenance enumeration, discovery, and assignment using only DESTID = RIO_BASE_ID (0x1060) or one of the other 15 device ID value. Supports infinite multicast/unicast groups No packet forwarding	1	1	0
F	Host can perform maintenance enumeration, discovery, and assignment using any DESTID. Allows hard coding (pin strapping) of BASE_ID, and then maintenance read back by the host. Supports infinite multicast/unicast group No packet forwarding	1	1	1

Mode A and Mode C are the legacy modes. Mode B is a superset of Mode A, and Mode F is a superset of Mode E. The most common modes are Mode C, Mode D, and Mode F.

The physical layer destID checking for Ftype 8 packets is controlled with f8_tgt_id_dis. This bit only disables maintenance packet checking at the physical layer. That is, if this bit is active, then regardless of the Ftype 8 packet's destID, the physical layer will accept and handle the maintenance request. If this bit is inactive, then the non-matching Ftype-8 packets are forwarded to logical layer, where it can be packet-forwarded or destroyed accordingly.

The src_tgt_id_dis bit is the legacy control bit for disabling destID checking on all other packet types. If this bit is active, then all packets regardless of destination ID are forwarded to the logical layer. If this bit is inactive, then the non-matching packets are destroyed before reaching the logical layer.

The new log_tgt_id_dis bit disables destID checking in the logical layer. If this bit is active and we are in mode E or F, then all packets regardless of the destID are forwarded to the appropriate functional block of the peripheral. If this bit is inactive, then all non-matching packets are destroyed. This bit allows promiscuous multicast and unicast IDs for all supported packet types, not just posted operations. This means that packet forwarding features can not be used.

Due to the current RTL implementation, there are some considerations when this bit is active:

- **MAU and RXU:** There are no side affects for request packets targeted for MAU. This means that if the log_tgt_id_dis bit is active, then all packets regardless of destination ID will be accepted and forwarded to the appropriate functional block.
- **RXU:** If the dest_prom is 0 for the mapping table, and there is no destID match, the request is discarded.
- **LSU:** Due to the current LSU implementation (LSU scoreboards the destID for response packets), those direct I/O response packets that are not targeted to the endpoint device is not accepted.
- **TXU:** Due to the current TXU implementation, DestID is in the CAM. Those message response packets that are not targeted to the endpoint device are not accepted.
- **CCU:** Due to the current CCU implementation (CCU does not verify the destID for their request packets), those congestion control packets that are not targeted to the endpoint device are accepted.

2.3.12.1 Discrete Multicast ID Support

The legacy multicast mode supports a single multicast ID that can be used for in-coming multicast requests. This support has been expanded to include 8 multicast IDs. In operation modes C and D, the allowable multicast deviceIDs are stored in the RIO_MULTIID_REG1, RIO_MULTIID_REG2, RIO_MULTIID_REG3, RIO_MULTIID_REG4, RIO_MULTIID_REG5, RIO_MULTIID_REG6, RIO_MULTIID_REG7, and RIO_MULTIID_REG8.

When a packet is received, the packet's tt field and DestID are checked against all the deviceIDs and the multicast IDs. If it does not match any and packet forwarding is disabled, the packet is destroyed and not forwarded to the protocol unit. If it matches one of these, it is forwarded to the relevant protocol unit. Because multicast operations are defined to be operations that do not require responses, they are limited to NWRITE, SWRITE, and type9 operations, and forwarded to the MAU as well as the RXU for type9 only. The multicast mode is disabled by simply writing the main deviceID (0x0080) into the multicast ID registers.

Multicast transactions are I/O packets that specify a destination memory address within the header. This address is used directly for the VBUSM transfer and is not modified in any way. For this reason, multicast support is limited to groups containing devices with the same memory map, or other devices that can perform address translation. It is the responsibility of the system designer to pre-determine valid multicast address ranges.

2.3.12.2 Promiscuous ID and DestID Support

Promiscuous ID and unicast DestIDs can be supported in modes E and F shown in Table 2-26. In these modes, all SUPPORTED packet types are accepted and sent to the appropriate peripheral functional blocks, regardless of incoming DestID. This means that non-posted transaction types such as NWRITE_R and messages requests can be sent to destIDs other than the BASE_ID value. When a response packet is sent out, the DestID and SourceID of the incoming request packet are swapped. Packet forwarding can not be used in these modes.

RapidIO now requires that devices support a promiscuous mode of operation for device discovery and enumeration. This means that devices should be able to respond to all incoming destIDs for ftype 8 maintenance after boot. Mode F supports this requirement and is more likely to be used.

2.3.13 Daisy-Chain Operation and Packet Forwarding

Some applications may require daisy chaining of devices together, instead of using a switch fabric. Typically, these applications are low-cost implementations. Daisy chains have variable system latency depending on device position within the chain. Daisy-chain implementations also have reduced bandwidth capabilities, as the link bandwidth doesn't change, the bandwidth allocated to each device in the chain is limited (sum of devices' individual bandwidth needs cannot exceed link bandwidth). To support daisy-chain or ring topologies, the peripheral features a hardware packet forwarding function. This feature eliminates the need for software to be involved in routing a packet to the next device in the chain. The basic idea behind the hardware packet forwarding logic is to provide an input port to output port path such that the packets never leave the peripheral (no DMA transfer). Mode C shown in Table 2-26 supports hardware packet forwarding. A simple check of an in-coming packet's DestID versus the device's DeviceID and MulticastIDs is done to determine if the packet should be forwarded. If the packet's DestID matches DeviceID, the packet is accepted and processed by the device. If the packet's DestID matches any MulticastID, the packet is accepted by the device and forwarded based on the rules outlined below. If the packet's DestID doesn't match either, the packet is simply destroyed or forwarded, depending on the hardware packet forwarding setup.

Additionally, it is beneficial to be able to only forward a packet if the destination ID is one of the devices in the chain/ring. Otherwise, a rogue packet may be forwarded, endlessly using up valuable bandwidth. The hardware packet forwarding uses a 4 entry mapping table. These mapping entries allow programmable selection of output port based on the in-coming packets DestID range.

The algorithm is as follows:

- The packet's DestID is compared against the mapping entries based on the packet's tt field. If tt = 2'b00, then the 8-bit version of ID boundaries is used. If tt = 2'b01, then the 16-bit version of ID boundaries is used.
- If any packet's DestID = DeviceID, it is handled normally and not forwarded. This is the highest priority check.
- If any packet's DestID = MulticastID, it is accepted. If the packet's DestID also falls into one of the ranges specified in the hardware packet forwarding mapping entries, the packet is also forwarded to the outbound port programmed.
- If the packet's DestID != DeviceID and DestID != MulticastID, but falls into one of the ranges specified in the hardware packet forwarding mapping entries, the packet is forwarded.
- If multiple table entry ranges are matched, then table entry 0 has highest priority, followed by table entry 1, and so forth.
- If the packet's DestID != DeviceID and DestID != MulticastID, and does not fall into one of the ranges specified in the hardware packet forwarding mapping entries, the packet is destroyed.
- Hardware packet forwarding can be disabled by assigning all the table entry Upper and Lower deviceID boundaries equal to the local DeviceID value.

Table 2-27 details the behavior of the hardware packet forwarding and multicast support, with respect to these specific packet types.

Table 2-27. RapidIO Packet Types

1	nread	(Ftype=2, Ttype=4'b0100)	**2**	atomic inc	(Ftype=2, Ttype=4'b1100)
3	atomic dec	(Ftype=2, Ttype=4'b1101)	**4**	atomic set	(Ftype=2, Ttype=4'b1110)
5	atomic clr	(Ftype=2, Ttype=4'b1111)	**6**	nwrite	(Ftype=5, Ttype=4'b0100)
7	nwrite_r	(Ftype=5, Ttype=4'b0101)	**8**	atomic t&s	(Ftype=5, Ttype=4'b1110)
9	swrite	(Ftype=6)	**10**	congestion	(Ftype=7)
11	maint read	(Ftype=8, Ttype=4'b0000)	**12**	maint write	(Ftype=8, Ttype=4'b0001)
13	maint rd resp	(Ftype=8, Ttype=4'b0010)	**14**	maint wr resp	(Ftype=8, Ttype=4'b0011)
15	maint port wr	(Ftype=8, Ttype=4'b0100)	**16**	doorbell	(Ftype=10)
17	message	(Ftype=11)	**18**	resp w/o payload	(Ftype=13, Ttype=4'b0000)
19	message resp	(Ftype=13, Ttype=4'b0001)	**20**	resp w/ payload	(Ftype=13, Ttype=4'b1000)
21	Type9 Traffic	(Ftype = 9)			

The numbers in Table 2-27 are referenced in Table 2-28.

Table 2-28. RapidIO Packet Type Behaviors

Incoming Packet DestID matches			Behavior
Local DeviceID	MulticastID	One of the packet forwarding table entry ranges	
Yes	N/A	N/A	All packet types 1 to 21 are handled accordingly by the local corresponding logical layer protocol unit (LSU, MAU, TXU, RXU, and CCU). Physical layer handles 11, 12, and 15.
No	Yes	No	All packet types 1 to 21 are forwarded to the MAU. The MAU only supports 1, 6, 7, 9, and 16. All other types are not supported and may cause undefined behavior, including the generation of an ERROR response. 21 is also forwarded to the RXU.
No	Yes	Yes	All packet types 1 to 21 are forwarded to the MAU. The MAU supports types 1, 6, 7, 9, and 16. All other types are not supported and may cause undefined behavior, including the generation of an ERROR response. The MAU will forward supported packet types by copying inbound to outbound. The port ID, specified in the packet forwarding table entry, is used for the outbound forwarded packet, as well as any outbound response packets. (1) 21 is also forwarded to the RXU.
No	No	Yes	All packet types 1 to 21 are forwarded to the MAU. The MAU copies the packet from inbound to outbound. The outbound port ID is specified by the packet forwarding table entry. *Note1 is valid for this case as well if the SRCID is also the same as the DESTID.
No	No	No	All packet types 1 to 21 are destroyed internally.

(1) Having responses packets transmitted from a different port than the incoming request packet is NOT standard RapidIO practice. If this is prohibited due to system topology, only packet types 6 and 9 should be used. Because the packet forwarding is done at the logical layer and not the physical layer, CRCs are regenerated for each forwarded packet.

Table 2-29. Hardware Packet Forwarding Mapping Entries Register Fields

Control/Command Register Field	Bits	R/W	Reset value (vbus_rst)	Description
DEVID_16b_LO	15-0	RW	16'hFFFF	Lower 16b DeviceID boundary. DestID lower than this number cannot use the table entry
DEVID_16b_UP	31-16	RW	16'hFFFF	Upper 16b DeviceID boundary. DestID above this range cannot use the table entry

Table 2-30. Hardware Packet Forwarding Mapping Entries Register Field

Bits	Control/Command Register Field	R/W	Reset value (vbus_rst)	Description
31-18	Reserved	RO	14'b0	Reserved
17-16	OUT_PORT	RW	2'b11	Output port number for packet's whose DestID falls within the 8b or 16b range for this table entry
15-8	DEVID_8b_UP	RW	8'hFF	Upper 8b DeviceID boundary. DestID above this range cannot use the table entry
7-0	DEVID_8b_LO	RW	8'hFF	Lower 8b DeviceID boundary. DestID lower than this number cannot use the table entry

- Packet Forwarding0 uses the registers PF_8b_CNTL0, PF_16b_CNTL0
- Packet Forwarding1 uses the registers PF_8b_CNTL1, PF_16b_CNTL1
- Packet Forwarding2 uses the registers PF_8b_CNTL2, PF_16b_CNTL2
- Packet Forwarding3 uses the registers PF_8b_CNTL3, PF_16b_CNTL3
- Packet Forwarding4 uses the registers PF_8b_CNTL4, PF_16b_CNTL4
- Packet Forwarding5 uses the registers PF_8b_CNTL5, PF_16b_CNTL5
- Packet Forwarding6 uses the registers PF_8b_CNTL6, PF_16b_CNTL6
- Packet Forwarding7 uses the registers PF_8b_CNTL7, PF_16b_CNTL7

2.3.14 Error Handling and Logging

Error Management registers allow detection and logging of physical layer and logical/transport layer errors. SP(n)_ERR_DET register illustrates the detectable physical layer (port) errors. ERR_DET illustrates the detectable logical/transport layer errors. The peripheral supports all detectable logical/transport errors except bits 29 and 26. The logical functional blocks involved for each detectable error condition are listed below, along with a brief description of the capture register contents.

Some errors are not captured by the error logging interface. Those incoming packets are simply dropped at the logical layer. This includes:

- If the TT bits are not legal
- No DeviceID, multicast or packet-forwarding match
- The incoming request is of an illegal FTYPE.
- The incoming request is for a unit that is not enabled yet.

If the physical layer is in the mode where it is checking the device IDs and the ID does not match, the incoming request is dropped.

Logical Layer Error Detect CSR:

- bit 31: LSU (request packet is logged)
- bit 30: TXU (request packet is logged)
- bit 29: Not supported
- bit 28: RXU (request packet is logged)
- bit 27: LSU and TXU (response packet is logged),
- MAU and RXU (request packet is logged)
- bit 26: Not supported
- bit 25: RXU (request packet is logged)
- bit 24: LSU and TXU (request packet is logged)
- bit 23: LSU and TXU (response packet is logged)
- bit 22: MAU (request packet is logged)
- bit 14: RXU (response packet is logged type9)
- bit 13: RXU (response packet is logged type9)
- bit 12: RXU (response packet is logged type9)
- bit 11: RXU (response packet is logged type9)
- bit 10: RXU (response packet is logged type9)
- bit 9: RXU (response packet is logged type9: No segmentation context was available for type9 traffic)
- bit 7: RXU (request packet is logged)
- bit 6: MAU (request packet is logged)

Table 2-31 lists the error response behavior of the logical layer with regards to the compliance checklist.

Table 2-31. Error Case Handling

Error	Intended SRIO_UA Behavior
Class 1	
NWRITE, SWRITE and NWRITE_R request packet data payload exceeds size specified in wrsize field	• Error Response [1] • Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
NWRITE, SWRITE and NWRITE_R request packet data payload exceeds 256 bytes	Physical layer destroys packet and sends Packet-not-accepted control symbol.
Received read request packet has a data payload	• Error response • Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Received NWRITE, SWRITE or NWRITE_R request packet has no data payload	• Error response [1] • Discard packet. • Set bit 27 of the Logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Received request packet uses a reserved field encoding for a required field	• Error response [1] • Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Received request packet uses illegal combinations of field encodings. For example, any ATOMIC transaction of more than 4 bytes.	• Error response [1] • Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write

(1) Error responses will not be sent for incoming NWRITE and SWRITE requests

Table 2-31. Error Case Handling (continued)

Error	Intended SRIO_UA Behavior
Unsupported transaction requested	• Error response (1) • Discard packet. • Set bit 22 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Received request packet makes use of an address that is not 8 byte aligned	The address field is always the 29MSBs of a 32b address.
Received NWRITE, NWRITE_R, SWRITE request data payload is not a multiple of 8 bytes.	• Error response (1) • Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Class 2	
Received unsolicited response packet	• Discard packet. • Set bit 23 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write (srvl_interface
NREAD “DONE” response packet data payload is more or less than requested data quantity	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write • LSU CC code = 011
NREAD “DONE” response packet has no data payload	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write LSU CC code = 011

Table 2-31. Error Case Handling (continued)

Error	Intended SRIO_UA Behavior
NREAD “DONE” response packet data payload is not a multiple of 8 bytes. Requests for sizes not a multiple of 8 bytes are padded to a multiple of 8 bytes.	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write LSU CC code = 011
Write response has a data payload	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write LSU CC code = 011 Allowed for test-and-swap type 5 packets.
Response packet time-out	• Set bit 24 of the logical/transport layer error detect CSR • Capture request: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set LSU CC = 001
Received “ERROR” response status	• Discard packet. • Set bit 31 of the logical/transport layer error detect CSR • Capture response: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set LSU CC = 011
“ERROR” response packet has a data payload	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture response: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set LSU CC=011

Table 2-31. Error Case Handling (continued)

Error	Intended SRIO_UA Behavior
Received response packet has reserved field encoding	• Discard packet. • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set LSU CC=011
NWRITE/NWRITE_R packet data size exceeds specified size.	LSU adheres to this rule for out-going requests. Thus this can never occur by design.
Requests must have addresses which are 8 byte aligned.	The address field is always the 29MSBs of a 32b address.
Class 3	
• 2A. DETAIL: Received MESSAGE packet has no data payload	• Discard packet. • Error response • Set bit 28 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
• 2B. DETAIL: MESSAGE transaction payload is not a multiple of 8 bytes in size	• Discard packet. • Error response • Set bit 28 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
• 2C. DETAIL: More than 16 individual MESSAGE transactions are used in one data message operation	From the receive side, any additional message segments would be interpreted as the next message to the same mailbox. RETRY based on a busy queue.
• 2D. DETAIL: msgseg value is greater than msglen value	• Discard packet. • Error response • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write

Table 2-31. Error Case Handling (continued)

Error	Intended SRIO_UA Behavior
• 2E. DETAIL: msgseg value is repeated and there was not a “RETRY” response for the previous usage	• Discard packet. • Error response • Set bit 27 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
• 2F. DETAIL: Received MESSAGE packet data payload is not of the size specified in the ssize field (exception for last packet which may be less)	• Discard packet. • Error response • Set bit 28 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set CC = 01
• 2G. DETAIL: Logical timeout on a message transaction is detected by the transmitter.	• Set bit 24 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set CC = 11
• 2H. DETAIL: All packets for the same message operation do not have the same mbox and letter fields.	On the RX side, this is impossible to test and should be removed.
RXU response-to-request timeout violation	• Set bit 25 of the logical/transport layer error detect CSR • Capture request: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set RXU CC = 010
ERROR Message Response	• Set bit 30 of the logical/transport layer error detect CSR • Capture request: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write Set TXU CC = 001

Table 2-31. Error Case Handling (continued)

Error	Intended SRIO_UA Behavior
Unsolicited Message Response	• Set bit 23 of the logical/transport layer error detect CSR • Capture request: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR • Initiate port write
Internal DMA error which prohibits completion of a DirectIO request	• Discard packet. • Error response (only if NREAD or NWRITE_R) • Set bit 6 of the logical/transport layer error detect CSR • Capture: – Logical/transport layer high address capture CSR – Logical/transport layer address capture CSR – Logical/transport layer device ID capture CSR – Logical/transport layer control capture CSR Initiate port write

If enabled (bit 0 of address offset 0x1B934), out-going port-writes are generated automatically by the SRIO peripheral under error conditions and sent to the port-write deviceID specified in 0xC028. There are both physical layer and logical layer errors that can generate the outgoing port-write. Descriptions for the detectable error conditions are found in offsets: 0xC008 and 0xC040, 0xC080, 0xC0C0, and 0xC100. The detectable errors must be individually enabled for reporting using 0xC00C, 0xC044, 0xC084, 0xC0C4, and 0xC104, respectively.

An incoming port-write is captured by the port-write capture registers: offset 0x1BA20-0x1BA2C and can cause an interrupt if enabled with bit 16 of 0x1B914, and bit 1 of 0x01E0 is routed to a CPU interrupt.

The contents of the error reporting port-write are:

- 0x1BA20 = Component TAG CSR,
- 0x1BA24 = Port N Error Detect CSR,
- 0x1BA28 = [31-14] RapidIO PLM Port {0..3} Event Status Register (bits 31-14);
  - [13-11] — Reserved
  - [10] - RCS bit of RapidIO Event Management Port-Write Status Register
  - [9] - MULTIPORT_ERR bit of RapidIO Event Management Port-Write Status Register
  - [8] - LOCALOG bit of RapidIO Event Management Port-Write Status Register
  - [7-0] - Port ID of the physical-layer event that triggered the Port-Write
- 0x1BA2C = RIO Logical/Transport Layer Error Detect CSR

A port-write can be generated with software for debug purposes by programming the LSU just as you would with any other maintenance packet transfer. The only difference is that a port-write only supports 16B payloads and that the config_offset is reserved for port-write.

2.3.15 Initialization Example

2.3.15.1 Enabling SRIO Peripheral

When the device is powered on, the SRIO peripheral is in a disabled state. Before any SRIO-specific initialization can take place, the peripheral must be enabled; otherwise, its registers cannot be written, and the reads will all return a value of zero.

```
/* Glb enable srio */
SRIO_REGS->GBL_EN = 0x00000001 ;
SRIO_REGS->BLK0_EN = 0x00000001 ; //MMR_EN
SRIO_REGS->BLK5_EN = 0x00000001 ; //PORT0_EN
SRIO_REGS->BLK1_EN = 0x00000001 ; //LSU_EN
SRIO_REGS->BLK2_EN = 0x00000001 ; //MAU_EN
SRIO_REGS->BLK3_EN = 0x00000001 ; //TXU_EN
SRIO_REGS->BLK4_EN = 0x00000001 ; //RXU_EN
SRIO_REGS->BLK6_EN = 0x00000001 ; //PORT1_EN
SRIO_REGS->BLK7_EN = 0x00000001 ; //PORT2_EN
SRIO_REGS->BLK8_EN = 0x00000001 ; //PORT3_EN
```

2.3.15.2 PLL, Ports, Device ID and Data Rate Initialization

To enable writing to read-only registers, the boot complete bit of the PER_SET_CNTL register (bit 24) must be set to 0.

Here is an example.

```
//  MAU_LEND_SWAP_MODE=0, LSU_LEND_SWAP_MODE=0, LOG_TGT_ID_DIS=1,
//  SOFTWARE_MEMORY_SLEEP_OVERRIDE=1, LOOPBACK=0, BOOT_COMPLETE=0,
//  TXU_RXU_LEND_SWAP_MODE=0, PROMOTE_DIS=0, TX_PRI2_WM=1,
//  TX_PRI1_WM=2, TX_PRI0_WM=3, CBA_TRANS_PRI=4, 1X_MODE=1,
//  PRESCALER_SELECT=6 (333.33MHz), SERDES3_PRBS_OVR=0,
//  SERDES2_PRBS_OVR=0, SERDES1_PRBS_OVR=0, SERDES0_PRBS_OVR=0
SRIO_REGS->PER_SET_CNTL = 32'h0C053960;
```

SerDes initialization is also part of this section. Refer to the SerDes initialization section for programming details.

2.3.15.3 Peripheral Initializations

```
// Set Device ID Registers
rdata = SRIO_REGS->DEVICEID_REG1;
wdata = 0x00ABBEEF;
mask = 0x00FFFFFF;
mdata = (wdata & mask) | (rdata & ~mask);
SRIO_REGS->DEVICEID_REG1 = mdata ; // id-16b=BEEF, id-08b=AB

rdata = SRIO_REGS->DEVICEID_REG2;
wdata = 0x00ABBEEF;
mask = 0x00FFFFFF;
mdata = (wdata & mask) | (rdata & ~mask);
SRIO_REGS->DEVICEID_REG2 = mdata ; // id-16b=BEEF, id-08b=AB

// Make sure BOOT_COMPLETE bit is cleared
rdata = SRIO_REGS->PER_SET_CNTL;
data = 0x00000000;
mask = 0x01000000;
mdata = (wdata & mask) | (rdata & ~mask);
SRIO_REGS->PER_SET_CNTL = mdata; // bootcmpl=0

SRIO_REGS->DEV_ID         = 32'hBEEF0030;  // id=BEEF, ti=0x0030
SRIO_REGS->DEV_INFO       = 32'h00000000;  // all 0
SRIO_REGS->ASBLY_ID       = 32'h00000030;  // ti=0x0030
SRIO_REGS->ASBLY_INFO     = 32'h00000100;  // 0x0000, next ext=0x0100
```

```
SRIO_REGS->PE_FEAT        = 32'h20000199;  // proc, bu ext, flow ctrl,
// retx suppress, 16-bit ID, 34-bit addr
SRIO_REGS->SRC_OP         = 32'h0004FDF4;  // all N:included data streaming
SRIO_REGS->DST_OP         = 32'h0000FC04;  // all except atomic
SRIO_REGS->PE_LL_CTL      = 32'h00000001;  // 34-bit addr
SRIO_REGS->LCL_CFG_HBAR   = 32'h00000000;  // all 0
SRIO_REGS->LCL_CFG_BAR    = 32'h00000000;  // all 0
SRIO_REGS->BASE_ID        = 32'h00ABBEEF;  // 16b-id=BEEF, 08b-id=AB
SRIO_REGS->H_BASE_ID_LOCK = 32'h0000BEEF;  // id=BEEF, lock
SRIO_REGS->SP_MB_HEAD      = 32'h10000002;  // end-point with sw recovery
SRIO_REGS->SP0_CTL        = 32'h00600001;  // enable i/o
SRIO_REGS->SP1_CTL        = 32'h00600001;  // enable i/o
SRIO_REGS->SP2_CTL        = 32'h00600001;  // enable i/o
SRIO_REGS->SP3_CTL        = 32'h00600001;  // enable i/o
SRIO_REGS->PLM_SP0_DISCOVERY_TIMER = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP1_DISCOVERY_TIMER = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP2_DISCOVERY_TIMER = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP3_DISCOVERY_TIMER = 32'h20000000;  // short cycles
SRIO_REGS->PRESCALAR_SRV_CLK           = 32'h00000021;  // 333MHz
SRIO_REGS->PLM_SP0_SILENCE_TIMER   = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP1_SILENCE_TIMER   = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP2_SILENCE_TIMER   = 32'h20000000;  // short cycles
SRIO_REGS->PLM_SP3_SILENCE_TIMER   = 32'h20000000;  // short cycles

// Set BOOT_COMPLETE bit
rdata = SRIO_REGS->PER_SET_CNTL;
wdata = 0x01000000;
mask = 0x01000000;
mdata = (wdata & mask) | (rdata & ~mask);
SRIO_REGS->PER_SET_CNTL = mdata; // bootcmpl=1

SRIO_REGS->SP_LT_CTL   = 32'h000FFF00;  // 839.5us
SRIO_REGS->SP_RT_CTL   = 32'hFFFFFF00;  // 839.5us
SRIO_REGS->SP_GEN_CTL  = 32'h40000000;  // agent, master, undiscovered
SRIO_REGS->ERR_RPT_BH  = 32'h30000007;  // next ext=0x0000(last)
SRIO_REGS->ERR_DET     = 32'h00000000;  // clear
SRIO_REGS->ERR_EN      = 32'h00000000;  // disable
SRIO_REGS->H_ADDR_CAPT = 32'h00000000;  // clear
SRIO_REGS->ADDR_CAPT   = 32'h00000000;  // clear
SRIO_REGS->ID_CAPT     = 32'h00000000;  // clear
SRIO_REGS->CTRL_CAPT   = 32'h00000000;  // clear
SRIO_REGS->PW_RX_CAPT0 = 32'h00000000;  // clear
SRIO_REGS->PW_RX_CAPT1 = 32'h00000000;  // clear
SRIO_REGS->PW_RX_CAPT2 = 32'h00000000;  // clear
SRIO_REGS->PW_RX_CAPT3 = 32'h00000000;  // clear


//INIT_WAIT - Wait for lane initialization


// Read register to check portx(1-4) OK bit (4 port example)
// polling SRIO_MAC's port_ok bit
rdata = SRIO_REGS->P0_ERR_STAT ;
while ((rdata & 0x00000002) != 0x00000002)
{
 rdata = SRIO_REGS->P0_ERR_STAT ;
}
if (srio4p1x_mode)
{
 rdata = SRIO_REGS->P1_ERR_STAT;
 while ((rdata & 0x00000002) != 0x00000002)
 {
  rdata = SRIO_REGS->P1_ERR_STAT;
 }
 rdata = SRIO_REGS->P2_ERR_STAT;
 while ((rdata & 0x00000002) != 0x00000002)
```

```
 {
  rdata = SRIO_REGS->P2_ERR_STAT;
 }
 rdata = SRIO_REGS->P3_ERR_STAT;
 while ((rdata & 0x00000002) != 0x00000002)
 {
 rdata = SRIO_REGS->P3_ERR_STAT;
 }
}

// Assert the PEREN bit to enable logical layer data flow
SRIO_REGS->PCR = 0x00000005; // peren
```

2.3.16 Optimization Techniques

This section describes optimization techniques for applications that use Serial RapidIO. Specifically, it targets applications meant to run on systems consisting of very high bandwidth transfers through SRIO, using very large packet sizes.

2.3.16.1 Overview

There are several parameters the user can control that affect the streamlining and optimization of SRIO packets sent throughout a system.

For Type 11 and Type 9 messages these include the following:

- Packet segmentation
- Packet DMA channel configuration
- Context allocation
- Receive flow configuration

For Type 11, Type 9, and DirectIO this includes the following:

- Priority scheduling

2.3.16.2 Packet Segmentation

When configuring descriptors for type 9 and type 11 operations, it is common to program the descriptors to use the largest possible packet length per message. Using the maximum packet length for transmit and receive descriptors reduces the amount of transactions that must occur for the packet DMA to construct the SRIO packets from the given descriptor information. This applies to both transmit and receive operations. The exception to this is if a single message requires spreaded buffers, in which case multiple descriptors would need to be linked together for a single message. If this is the case, the user should keep the packet length a multiple of 2 to minimize the number of requests the packet DMA must perform to form the packets from the buffer memory. A multiple of 2 length ensures that the 64-byte bursts used to retrieve data from memory and store it into the prefetch FIFO are used efficiently. In addition, regardless of the size of the packet payload, the packet DMA buffer memory consumes a full 256-byte buffer space when sending or receiving a packet.

The user can also control how the hardware segments the outgoing message through the segment size information in the transmit descriptor. Regardless of the packet length configured in the descriptor, the SRIO logical layer may divide the final message into smaller segments (the maximum segment it can pass is 256B). Use the maximum possible segment size to take full advantage of the space available in the outbound credit buffers. The reason for this is that regardless of the segment size used, each segment destined for the outbound credit buffers will take up the full 256 byte chunks that the buffers are partitioned into. Thus, if the packets are larger than 256B, then use a 256B segment size. If the packets are smaller than 256B, than it may be more beneficial to tailor the segment size to the individual packet size. For instance, if the packet sizes are 64B, a segment size of 64B would be optimal.

2.3.16.3 Packet DMA Channel Configuration

Bandwidth utilization within the packet DMA can be increased when using multiple packet DMA channels. Using multiple channels allows multiple send and receive transactions to occur in parallel within the packet DMA subsystem. Enable all available SRIO packet DMA channels for both transmit and receive to spread the load of the traffic within the packet DMA. When configuring transmit descriptors, do not link the descriptors unless required for scattered operations (the receive side is linked automatically). Rather, ensure that the descriptors are pushed as evenly as possible amongst the 16 available transmit queues tied to the 16 transmit channels. As long as the priorities are set to the same value for all the channels, they will be arbitrated in round-robin fashion. For receive operations, configuring multiple receive queues tied to separate receive channels also helps optimize bandwidth internally within the packet DMA.

2.3.16.4 Context Allocation

The device-specific information can be configured to actively allow a specific number of segmentation contexts to be used on the transmit and receive side. Enabling all segmentation contexts allows the device to simultaneously track the maximum number of message streams.

2.3.16.5 Receive Flow Configuration

SRIO has 32 receive flows available, which can be tied to specific queues. Regardless of the number of queues configured, the hardware always utilizes all of the receive flows as long as they are configured. This allows an increase in the processing on the receive side in relation to the packet DMA.

2.3.16.6 Priority Scheduling

Another way to improve bandwidth is by avoiding mixed packet sizes, which may increase the amount of time it takes to fill up the various buffers. In a scenario where many large packets are received, the logical-layer shared buffers can be consumed more quickly, because it takes longer for the data to be transferred to the different protocol units. In this time with the shared buffers occupied, if a large number of small packets are received behind the large packets, the physical layer buffers can become backed up. In the case of type 11, this behavior could eventually lead to the flow control having to send RETRYs for packets that don't make it into the physical layer buffers.

In the case of type 9 and type 11, if it is necessary to use varying packet sizes, configure the transmit queues with different priorities and dedicate different packet sizes to different transmit channels. This minimizes the interleaving of different sizes.

2.3.16.7 Transmit and Receive Operations

These optimization techniques can be summarized by source of operation as described in the following sections (applies only to type 9 and type 11 unless specified).

2.3.16.7.1 Transmit Operations

1. Use the largest possible packet length for descriptors. If this is not possible, keep the size to a multiple of 2.
2. Use the maximum segment size (256B) in transmitting protocol-specific information, unless the entire message is less than 256B.
3. Avoid linking transmit descriptors unless truly required. Rather, spread descriptors evenly amongst as many transmit queues as possible to utilize more packet DMA channels simultaneously.
4. Enable all transmit contexts.
5. Avoid sending mixed packet sizes out on the same priority (applies to all packet types including DirectIO).

2.3.16.7.2 Receive Operations

1. Use the largest possible packet length for descriptors. If this is not possible, keep the size to a multiple of 2.
2. Use as many receive packet DMA receive channels as possible, by using multiple receive queues.
3. Enable all receive contexts.
4. Enable all receive flows.

SRIO Registers

Table 3-1 lists the names and address offsets of the memory-mapped registers for the Serial RapidIO (SRIO) peripheral. See the specific-device data sheet for the exact memory addresses of these registers. Registers with address offset less than 1000h are DSP peripheral control and status registers. Registers above 1000h are RapidIO-specific registers. The collection of RapidIO-specific registers are detailed within the RapidIO Interconnect Specification, LP-Serial Physical Layer specification, and the Error Management Extensions Specification.

RapidIO terminology refers to two types of registers: Capability Registers (CAR) and Command and Status Registers (CSR). These registers allow an external processing element to determine another processing element's capabilities, as well as control and determine the status of its internal hardware. Thus, these registers are accessible from an external processing element. These registers are accessed through the I/O logical maintenance operation using maintenance packets. Maintenance packets specify the Config_offset (21-bit) of the CAR/CSR to be accessed. The RapidIO configuration space is based on 0x0-to-0xFFFFF8 offsets, and the Config_offset reflects those offsets in the maintenance packets. So, if the maintenance packet specifies the register offset to be 0x0000, it will be at offset 0x1000. Accesses to non-CAR/CSR registers, such as the DSP Peripheral Control/Status registers, are NOT allowed through maintenance packets. DSP accesses to the RapidIO CAR/CSR space must be adjusted by the 0x1000 offset. All registers are 32 bits wide and accessed in 32-bit (4 bytes) quantities from the RapidIO maintenance packets. By accesses from the DSP side is allowed through the configuration bus (VBUSP).

Topic **Page**

3.1 SRIO Register Offsets

Below is the overview of the SRIO register offsets:

Table 3-1. SRIO Register Offsets

Offset	Register
SRIO SERDES REGISTERS	
	SerDes Macro Registers
- 0x02620380	SerDes Receive/Transmit Channel Configuration Registers
SRIO REGISTERS	
- 0010h	Required Peripheral Registers
- 0018h	Peripheral Setting Control Registers
- 0078h	Global and Block Enable Registers
007Ch - 00DCh	ID Registers
- 0118h	Hardware Packet Forwarding Registers
- 0310h	Interrupt Registers
- 09FCh	RXU Registers
- 0E4Ch	LSU/MAU Registers
- 0EACh	Flow Control Registers
- 0EFCh	TXU Registers
- 2FFCh	CDMAHP Registers
- B1BCh	RIO CSR/CAR Registers
+	Error Management Registers

Table 3-2 shows the memory map for the registers.

Table 3-2. SRIO Registers

Offset	Acronym	Register Description	Section
SERDES MACRO REGISTERS			
0x02620154	SRIO_SERDES_STS	SerDes Macro Status Register	Section 3.2.1
0x02620360	SRIO_SERDES_CFGPLL	SerDes Macro Configuration Register	Section 3.2.2
SERDES RECEIVE/TRANSMIT CHANNEL CONFIGURATION REGISTERS			
0x02620364	SRIO_SERDES_CFGRX0	SerDes Receive Channel Configuration Register 0	Section 3.3.1
0x02620368	SRIO_SERDES_CFGTX0	SerDes Transmit Channel Configuration Register 0	Section 3.3.2
0x0262036C	SRIO_SERDES_CFGRX1	SerDes Receive Channel Configuration Register 1	Section 3.3.1
0x02620370	SRIO_SERDES_CFGTX1	SerDes Transmit Channel Configuration Register 1	Section 3.3.2
0x02620374	SRIO_SERDES_CFGRX2	SerDes Receive Channel Configuration Register 2	Section 3.3.1
0x02620378	SRIO_SERDES_CFGTX2	SerDes Transmit Channel Configuration Register 2	Section 3.3.2
0x0262037C	SRIO_SERDES_CFGRX3	SerDes Receive Channel Configuration Register 3	Section 3.3.1
0x02620380	SRIO_SERDES_CFGTX3	SerDes Transmit Channel Configuration Register 3	Section 3.3.2
REQUIRED PERIPHERAL REGISTERS			
0000h	PID	Peripheral Identification Register	Section 3.4.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0004h	PCR	Peripheral Control Register	Section 3.4.2
PERIPHERAL SETTING CONTROL REGISTERS			
0014h	PER_SET_CNTL	Peripheral Settings Control Reg	Section 3.5.1
0018h	PER_SET_CNTL1	Peripheral Settings Control Reg1	Section 3.5.2
GLOBAL & BLOCK ENABLE REGISTERS			
0024h	GBL_EN	Peripheral Global Enable	Section 3.6
0028h	GBL_EN_STAT	Peripheral Global Enable Status	Section 3.6
002Ch	BLK0_EN	Block0 Enable	Section 3.6
0030h	BLK0_EN_STAT	Block0 Enable Status	Section 3.6
0034h	BLK1_EN	Block1 Enable	Section 3.6
0038h	BLK1_EN_STAT	Block1 Enable Status	Section 3.6
003Ch	BLK2_EN	Block2 Enable	Section 3.6
0040h	BLK2_EN_STAT	Block2 Enable Status	Section 3.6
0044h	BLK3_EN	Block3 Enable	Section 3.6
0048h	BLK3_EN_STAT	Block3 Enable Status	Section 3.6
004Ch	BLK4_EN	Block4 Enable	Section 3.6
0050h	BLK4_EN_STAT	Block4 Enable Status	Section 3.6
0054h	BLK5_EN	Block5 Enable	Section 3.6
0058h	BLK5_EN_STAT	Block5 Enable Status	Section 3.6
005Ch	BLK6_EN	Block6 Enable	Section 3.6
0060h	BLK6_EN_STAT	Block6 Enable Status	Section 3.6
0064h	BLK7_EN	Block7 Enable	Section 3.6
0068h	BLK7_EN_STAT	Block7 Enable Status	Section 3.6
006Ch	BLK8_EN	Block8 Enable	Section 3.6
0070h	BLK8_EN_STAT	Block8 Enable Status	Section 3.6
0074h	BLK9_EN	Block9 Enable	Section 3.6
0078h	BLK9_EN_STAT	Block9 Enable Status	Section 3.6
ID REGISTERS			
0080h -00BCh	Reserved	Reserved	Section 3.7.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
00C0h	MULTIID_REG1	RapidIO Multicast ID	Section 3.7.1
00C4h	MULTIID_REG2	RapidIO Multicast ID	Section 3.7.2
00C8h	MULTIID_REG3	RapidIO Multicast ID	Section 3.7.2
00CCh	MULTIID_REG4	RapidIO Multicast ID	Section 3.7.2
00D0h	MULTIID_REG5	RapidIO Multicast ID	Section 3.7.2
00D4h	MULTIID_REG6	RapidIO Multicast ID	Section 3.7.2
00D8h	MULTIID_REG7	RapidIO Multicast ID	Section 3.7.2
00DCh	MULTIID_REG8	RapidIO Multicast ID	Section 3.7.2
HARDWARE PACKET FORWARDING REGISTERS			
00E0h	PF_16b_CNTL0	Packet Forwarding Reg 0 for 16b DeviceIDs	Section 3.8.1
00E4h	PF_8b_CNTL0	Packet Forwarding Reg 0 for 8b DeviceIDs	Section 3.8.2
00E8h	PF_16b_CNTL1	Packet Forwarding Reg 1 for 16b DeviceIDs	Section 3.8.1
00ECh	PF_8b_CNTL1	Packet Forwarding Reg 1 for 8b DeviceIDs	Section 3.8.2
00F0h	PF_16b_CNTL2	Packet Forwarding Reg 2 for 16b DeviceIDs	Section 3.8.1
00F4h	PF_8b_CNTL2	Packet Forwarding Reg 2 for 8b DeviceIDs	Section 3.8.2
00F8h	PF_16b_CNTL3	Packet Forwarding Reg 3 for 16b DeviceIDs	Section 3.8.1
00FCh	PF_8b_CNTL3	Packet Forwarding Reg 3 for 8b DeviceIDs	Section 3.8.2
0100h	PF_16b_CNTL4	Packet Forwarding Reg 4 for 16b DeviceIDs	Section 3.8.1
0104h	PF_8b_CNTL4	Packet Forwarding Reg 4 for 8b DeviceIDs	Section 3.8.2
0108h	PF_16b_CNTL5	Packet Forwarding Reg 5 for 16b DeviceIDs	Section 3.8.1
010Ch	PF_8b_CNTL5	Packet Forwarding Reg 5 for 8b DeviceIDs	Section 3.8.2
0110h	PF_16b_CNTL6	Packet Forwarding Reg 6 for 16b DeviceIDs	Section 3.8.1
0114h	PF_8b_CNTL6	Packet Forwarding Reg 6 for 8b DeviceIDs	Section 3.8.2
0118h	PF_16b_CNTL7	Packet Forwarding Reg 7 for 16b DeviceIDs	Section 3.8.1
011Ch	PF_8b_CNTL7	Packet Forwarding Reg 7 for 8b DeviceIDs	Section 3.8.2
0160h	Reserved (ERR_DET shadow register)	Logical/Transport Layer Error Detect CSR	Section 3.18.76
0164h	Reserved (ERR_EN Shadow Register)	Logical/Transport Layer Error Enable CSR	Section 3.18.77
0168h	Reserved (H_ADDR_CAPT Shadow Register)	Logical/Transport Layer High Address Capture CSR	Section 3.18.78

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
016Ch	Reserved (ADDR_CAPT Shadow Register)	Logical/Transport Layer Address Capture CSR	Section 3.18.79
0170h	Reserved (ID_CAPT Shadow Register)	Logical/Transport Layer Device ID Capture CSR	Section 3.18.80
0174h	Reserved (CTRL_CAPT Shadow Register)	Logical/Transport Layer Control Capture CSR	Section 3.18.81
INTERRUPT REGISTERS			
0180h	DOORBELL0_ICSR	DOORBELL0 Interrupt Status Reg	Section 3.9.1
0188h	DOORBELL0_ICCR	DOORBELL0 Interrupt Clear Reg	Section 3.9.2
0190h	DOORBELL1_ICSR	DOORBELL1 Interrupt Status Reg	Section 3.9.1
0198h	DOORBELL1_ICCR	DOORBELL1 Interrupt Clear Reg	Section 3.9.2
01A0h	DOORBELL2_ICSR	DOORBELL2 Interrupt Status Reg	Section 3.9.1
01A8h	DOORBELL2_ICCR	DOORBELL2 Interrupt Clear Reg	Section 3.9.2
01B0h	DOORBELL3_ICSR	DOORBELL3 Interrupt Status Reg	Section 3.9.1
01B8h	DOORBELL3_ICCR	DOORBELL3 Interrupt Clear Reg	Section 3.9.2
01C0h	LSU0_ICSR	LSU Status Int Reg	Section 3.9.3
01C8h	LSU0_ICCR	LSU Clear Int Reg	Section 3.9.4
01D0h	LSU1_ICSR	LSU Status Int Reg	Section 3.9.3
01D8h	LSU1_ICCR	LSU Clear Int Reg	Section 3.9.4
01E0h	ERR_RST_EVNT_ICSR	Error, Reset, and Special Event Status Int Reg	Section 3.9.5
01E8h	ERR_RST_EVNT_ICCR	Error, Reset, and Special Event Clear Int Reg	Section 3.9.5
0200h	DOORBELL0_ICRR	DOORBELL0 Interrupt Condition Routing Reg	Section 3.9.6
0204h	DOORBELL0_ICRR2	DOORBELL0 Interrupt Condition Routing Reg	Section 3.9.6
020Ch	DOORBELL1_ICRR	DOORBELL1 Interrupt Condition Routing Reg	Section 3.9.6
0210h	DOORBELL1_ICRR2	DOORBELL1 Interrupt Condition Routing Reg	Section 3.9.6
0218h	DOORBELL2_ICRR	DOORBELL2 Interrupt Condition Routing Reg	Section 3.9.6
021Ch	DOORBELL2_ICRR2	DOORBELL2 Interrupt Condition Routing Reg	Section 3.9.6
0224h	DOORBELL3_ICRR	DOORBELL3 Interrupt Condition Routing Reg	Section 3.9.6
0228h	DOORBELL3_ICRR2	DOORBELL3 Interrupt Condition Routing Reg	Section 3.9.6
0230h	LSU0_ICRR1	LSU Module Int Condition Routing Reg	Section 3.9.7
0234h	LSU0_ICRR2	LSU Module Int Condition Routing Reg	Section 3.9.7

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0238h	LSU0_ICRR3	LSU Module Int Condition Routing Reg	Section 3.9.7
023Ch	LSU0_ICRR4	LSU Module Int Condition Routing Reg	Section 3.9.7
0240h	LSU1_ICRR1	LSU1 Module Int Condition Routing Reg	Section 3.9.7
0250h	ERR_RST_EVNT_ICRR	Err/Reset/special event Int Condition Routing Reg	Section 3.9.8
0254h	ERR_RST_EVNT_ICRR2	Err/Reset/special event Int Condition Routing Reg	Section 3.9.8
0258h	ERR_RST_EVNT_ICRR3	Err/Reset/special event Int Condition Routing Reg	Section 3.9.8
0264h	INTERRUPT_CTL	Doorbell interrupts mapped to new 8 interrupts versus the legacy 16 interrupts	Section 3.9.9
0270h	INTDST0_Decode	INTDST0 Interrupt Status Decode Reg	Section 3.9.9.1
0274h	INTDST1_Decode	INTDST1 Interrupt Status Decode Reg	Section 3.9.9.1
0278h	INTDST2_Decode	INTDST2 Interrupt Status Decode Reg	Section 3.9.9.1
027Ch	INTDST3_Decode	INTDST3 Interrupt Status Decode Reg	Section 3.9.9.1
0280h	INTDST4_Decode	INTDST4 Interrupt Status Decode Reg	Section 3.9.9.1
0284h	INTDST5_Decode	INTDST5 Interrupt Status Decode Reg	Section 3.9.9.1
0288h	INTDST6_Decode	INTDST6 Interrupt Status Decode Reg	Section 3.9.9.1
028Ch	INTDST7_Decode	INTDST7 Interrupt Status Decode Reg	Section 3.9.9.1
0290h	INTDST8_Decode	INTDST8 Interrupt Status Decode Reg	Section 3.9.9.1
0294h	INTDST9_Decode	INTDST9 Interrupt Status Decode Reg	Section 3.9.9.1
0298h	INTDST10_Decode	INTDST10 Interrupt Status Decode Reg	Section 3.9.9.1
029Ch	INTDST11_Decode	INTDST11 Interrupt Status Decode Reg	Section 3.9.9.1
02A0h	INTDST12_Decode	INTDST12 Interrupt Status Decode Reg	Section 3.9.9.1
02A4h	INTDST13_Decode	INTDST13 Interrupt Status Decode Reg	Section 3.9.9.1
02A8h	INTDST14_Decode	INTDST14 Interrupt Status Decode Reg	Section 3.9.9.1
02ACh	INTDST15_Decode	INTDST15 Interrupt Status Decode Reg	Section 3.9.9.1
02B0h	INTDST16_Decode	INTDST16 Interrupt Status Decode Reg	Section 3.9.9.1
02B4h	INTDST17_Decode	INTDST17 Interrupt Status Decode Reg	Section 3.9.9.1
02B8h	INTDST18_Decode	INTDST18 Interrupt Status Decode Reg	Section 3.9.9.1
02BCh	INTDST19_Decode	INTDST19 Interrupt Status Decode Reg	Section 3.9.9.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
02C0h	INTDST20_Decode	INTDST20 Interrupt Status Decode Reg	Section 3.9.9.1
02C4h	INTDST21_Decode	INTDST21 Interrupt Status Decode Reg	Section 3.9.9.1
02C8h	INTDST22_Decode	INTDST22 Interrupt Status Decode Reg	Section 3.9.9.1
02CCh	INTDST23_Decode	INTDST23 Interrupt Status Decode Reg	Section 3.9.9.1
02D0h	INTDST0_Rate_CNTL	INTDST0 Interrupt Rate Control Reg	Section 3.9.9.2
02D4h	INTDST1_Rate_CNTL	INTDST1 Interrupt Rate Control Reg	Section 3.9.9.2
02D8h	INTDST2_Rate_CNTL	INTDST2 Interrupt Rate Control Reg	Section 3.9.9.2
02DCh	INTDST3_Rate_CNTL	INTDST3 Interrupt Rate Control Reg	Section 3.9.9.2
02E0h	INTDST4_Rate_CNTL	INTDST4 Interrupt Rate Control Reg	Section 3.9.9.2
02E4h	INTDST5_Rate_CNTL	INTDST5 Interrupt Rate Control Reg	Section 3.9.9.2
02E8h	INTDST6_Rate_CNTL	INTDST6 Interrupt Rate Control Reg	Section 3.9.9.2
02ECh	INTDST7_Rate_CNTL	INTDST7 Interrupt Rate Control Reg	Section 3.9.9.2
02F0h	INTDST8_Rate_CNTL	INTDST8 Interrupt Rate Control Reg	Section 3.9.9.2
02F4h	INTDST9_Rate_CNTL	INTDST9 Interrupt Rate Control Reg	Section 3.9.9.2
02F8h	INTDST10_Rate_CNTL	INTDST10 Interrupt Rate Control Reg	Section 3.9.9.2
02FCh	INTDST11_Rate_CNTL	INTDST11 Interrupt Rate Control Reg	Section 3.9.9.2
0300h	INTDST12_Rate_CNTL	INTDST12 Interrupt Rate Control Reg	Section 3.9.9.2
0304h	INTDST13_Rate_CNTL	INTDST13 Interrupt Rate Control Reg	Section 3.9.9.2
0308h	INTDST14_Rate_CNTL	INTDST14 Interrupt Rate Control Reg	Section 3.9.9.2
030Ch	INTDST15_Rate_CNTL	INTDST15 Interrupt Rate Control Reg	Section 3.9.9.2
0310h	INTDST_RATE_DIS	INTDST Pacing Disable	
RXU REGISTERS			
0400h	RXU_MAP00_L	MailBox-to-queue mapping register	Section 3.10.1.1
0404h	RXU_MAP00_H	MailBox-to-queue mapping register	Section 3.10.1.1
0408h	RXU_MAP00_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
040Ch	RXU_MAP01_L	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0410h	RXU_MAP01_H	MailBox-to-queue mapping register	Section 3.10.1.1
0414h	RXU_MAP01_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0418h	RXU_MAP02_L	MailBox-to-queue mapping register	Section 3.10.1.1
041Ch	RXU_MAP02_H	MailBox-to-queue mapping register	Section 3.10.1.1
0420h	RXU_MAP02_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0424h	RXU_MAP03_L	MailBox-to-queue mapping register	Section 3.10.1.1
0428h	RXU_MAP03_H	MailBox-to-queue mapping register	Section 3.10.1.1
042Ch	RXU_MAP03_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0430h	RXU_MAP04_L	MailBox-to-queue mapping register	Section 3.10.1.1
0434h	RXU_MAP04_H	MailBox-to-queue mapping register	Section 3.10.1.1
0438h	RXU_MAP04_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
043Ch	RXU_MAP05_L	MailBox-to-queue mapping register	Section 3.10.1.1
0440h	RXU_MAP05_H	MailBox-to-queue mapping register	Section 3.10.1.1
0444h	RXU_MAP05_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0448h	RXU_MAP06_L	MailBox-to-queue mapping register	Section 3.10.1.1
044Ch	RXU_MAP06_H	MailBox-to-queue mapping register	Section 3.10.1.1
0450h	RXU_MAP06_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0454h	RXU_MAP07_L	MailBox-to-queue mapping register	Section 3.10.1.1
0458h	RXU_MAP07_H	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
045Ch	RXU_MAP07_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0460h	RXU_MAP08_L	MailBox-to-queue mapping register	Section 3.10.1.1
0464h	RXU_MAP08_H	MailBox-to-queue mapping register	Section 3.10.1.1
0468h	RXU_MAP08_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
046Ch	RXU_MAP09_L	MailBox-to-queue mapping register	Section 3.10.1.1
0470h	RXU_MAP09_H	MailBox-to-queue mapping register	Section 3.10.1.1
0474h	RXU_MAP09_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0478h	RXU_MAP10_L	MailBox-to-queue mapping register	Section 3.10.1.1
047Ch	RXU_MAP10_H	MailBox-to-queue mapping register	Section 3.10.1.1
0480h	RXU_MAP10_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0484h	RXU_MAP11_L	MailBox-to-queue mapping register	Section 3.10.1.1
0488h	RXU_MAP11_H	MailBox-to-queue mapping register	Section 3.10.1.1
048Ch	RXU_MAP11_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0490h	RXU_MAP12_L	MailBox-to-queue mapping register	Section 3.10.1.1
0494h	RXU_MAP12_H	MailBox-to-queue mapping register	Section 3.10.1.1
0498h	RXU_MAP12_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
049Ch	RXU_MAP13_L	MailBox-to-queue mapping register	Section 3.10.1.1
04A0h	RXU_MAP13_H	MailBox-to-queue mapping register	Section 3.10.1.1
04A4h	RXU_MAP13_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
04A8h	RXU_MAP14_L	MailBox-to-queue mapping register	Section 3.10.1.1
04ACh	RXU_MAP14_H	MailBox-to-queue mapping register	Section 3.10.1.1
04B0h	RXU_MAP14_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04B4h	RXU_MAP15_L	MailBox-to-queue mapping register	Section 3.10.1.1
04B8h	RXU_MAP15_H	MailBox-to-queue mapping register	Section 3.10.1.1
04BCh	RXU_MAP15_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04C0h	RXU_MAP16_L	MailBox-to-queue mapping register	Section 3.10.1.1
04C4h	RXU_MAP16_H	MailBox-to-queue mapping register	Section 3.10.1.1
04C8h	RXU_MAP16_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04CCh	RXU_MAP17_L	MailBox-to-queue mapping register	Section 3.10.1.1
04D0h	RXU_MAP17_H	MailBox-to-queue mapping register	Section 3.10.1.1
04D4h	RXU_MAP17_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04D8h	RXU_MAP18_L	MailBox-to-queue mapping register	Section 3.10.1.1
04DCh	RXU_MAP18_H	MailBox-to-queue mapping register	Section 3.10.1.1
04E0h	RXU_MAP18_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04E4h	RXU_MAP19_L	MailBox-to-queue mapping register	Section 3.10.1.1
04E8h	RXU_MAP19_H	MailBox-to-queue mapping register	Section 3.10.1.1
04ECh	RXU_MAP19_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04F0h	RXU_MAP20_L	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
04F4h	RXU_MAP20_H	MailBox-to-queue mapping register	Section 3.10.1.1
04F8h	RXU_MAP20_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
04FCh	RXU_MAP21_L	MailBox-to-queue mapping register	Section 3.10.1.1
0500h	RXU_MAP21_H	MailBox-to-queue mapping register	Section 3.10.1.1
0504h	RXU_MAP21_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0508h	RXU_MAP22_L	MailBox-to-queue mapping register	Section 3.10.1.1
050Ch	RXU_MAP22_H	MailBox-to-queue mapping register	Section 3.10.1.1
0510h	RXU_MAP22_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0514h	RXU_MAP23_L	MailBox-to-queue mapping register	Section 3.10.1.1
0518h	RXU_MAP23_H	MailBox-to-queue mapping register	Section 3.10.1.1
051Ch	RXU_MAP23_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0520h	RXU_MAP24_L	MailBox-to-queue mapping register	Section 3.10.1.1
0524h	RXU_MAP24_H	MailBox-to-queue mapping register	Section 3.10.1.1
0528h	RXU_MAP24_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
052Ch	RXU_MAP25_L	MailBox-to-queue mapping register	Section 3.10.1.1
0530h	RXU_MAP25_H	MailBox-to-queue mapping register	Section 3.10.1.1
0534h	RXU_MAP25_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0538h	RXU_MAP26_L	MailBox-to-queue mapping register	Section 3.10.1.1
053Ch	RXU_MAP26_H	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0540h	RXU_MAP26_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0544h	RXU_MAP27_L	MailBox-to-queue mapping register	Section 3.10.1.1
0548h	RXU_MAP27_H	MailBox-to-queue mapping register	Section 3.10.1.1
054Ch	RXU_MAP27_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0550h	RXU_MAP28_L	MailBox-to-queue mapping register	Section 3.10.1.1
0554h	RXU_MAP28_H	MailBox-to-queue mapping register	Section 3.10.1.1
0558h	RXU_MAP28_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
055Ch	RXU_MAP29_L	MailBox-to-queue mapping register	Section 3.10.1.1
0560h	RXU_MAP29_H	MailBox-to-queue mapping register	Section 3.10.1.1
0564h	RXU_MAP29_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0568h	RXU_MAP30_L	MailBox-to-queue mapping register	Section 3.10.1.1
056Ch	RXU_MAP30_H	MailBox-to-queue mapping register	Section 3.10.1.1
0570h	RXU_MAP30_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0574h	RXU_MAP31_L	MailBox-to-queue mapping register	Section 3.10.1.1
0578h	RXU_MAP31_H	MailBox-to-queue mapping register	Section 3.10.1.1
057Ch	RXU_MAP31_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0580h	RXU_MAP32_L	MailBox-to-queue mapping register	Section 3.10.1.1
0584h	RXU_MAP32_H	MailBox-to-queue mapping register	Section 3.10.1.1
0588h	RXU_MAP32_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
058Ch	RXU_MAP33_L	MailBox-to-queue mapping register	Section 3.10.1.1
0590h	RXU_MAP33_H	MailBox-to-queue mapping register	Section 3.10.1.1
0594h	RXU_MAP33_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0598h	RXU_MAP34_L	MailBox-to-queue mapping register	Section 3.10.1.1
059Ch	RXU_MAP34_H	MailBox-to-queue mapping register	Section 3.10.1.1
05A0h	RXU_MAP34_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05A4h	RXU_MAP35_L	MailBox-to-queue mapping register	Section 3.10.1.1
05A8h	RXU_MAP35_H	MailBox-to-queue mapping register	Section 3.10.1.1
05ACh	RXU_MAP35_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05B0h	RXU_MAP36_L	MailBox-to-queue mapping register	Section 3.10.1.1
05B4h	RXU_MAP36_H	MailBox-to-queue mapping register	Section 3.10.1.1
05B8h	RXU_MAP36_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05BCh	RXU_MAP37_L	MailBox-to-queue mapping register	Section 3.10.1.1
05C0h	RXU_MAP37_H	MailBox-to-queue mapping register	Section 3.10.1.1
05C4h	RXU_MAP37_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05C8h	RXU_MAP38_L	MailBox-to-queue mapping register	Section 3.10.1.1
05CCh	RXU_MAP38_H	MailBox-to-queue mapping register	Section 3.10.1.1
05D0h	RXU_MAP38_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05D4h	RXU_MAP39_L	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
05D8h	RXU_MAP39_H	MailBox-to-queue mapping register	Section 3.10.1.1
05DCh	RXU_MAP39_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05E0h	RXU_MAP40_L	MailBox-to-queue mapping register	Section 3.10.1.1
05E4h	RXU_MAP40_H	MailBox-to-queue mapping register	Section 3.10.1.1
05E8h	RXU_MAP40_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05ECh	RXU_MAP41_L	MailBox-to-queue mapping register	Section 3.10.1.1
05F0h	RXU_MAP41_H	MailBox-to-queue mapping register	Section 3.10.1.1
05F4h	RXU_MAP41_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
05F8h	RXU_MAP42_L	MailBox-to-queue mapping register	Section 3.10.1.1
05FCh	RXU_MAP42_H	MailBox-to-queue mapping register	Section 3.10.1.1
0600h	RXU_MAP42_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0604h	RXU_MAP43_L	MailBox-to-queue mapping register	Section 3.10.1.1
0608h	RXU_MAP43_H	MailBox-to-queue mapping register	Section 3.10.1.1
060Ch	RXU_MAP43_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0610h	RXU_MAP44_L	MailBox-to-queue mapping register	Section 3.10.1.1
0614h	RXU_MAP44_H	MailBox-to-queue mapping register	Section 3.10.1.1
0618h	RXU_MAP44_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
061Ch	RXU_MAP45_L	MailBox-to-queue mapping register	Section 3.10.1.1
0620h	RXU_MAP45_H	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0624h	RXU_MAP45_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0628h	RXU_MAP46_L	MailBox-to-queue mapping register	Section 3.10.1.1
062Ch	RXU_MAP46_H	MailBox-to-queue mapping register	Section 3.10.1.1
0630h	RXU_MAP46_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0634h	RXU_MAP47_L	MailBox-to-queue mapping register	Section 3.10.1.1
0638h	RXU_MAP47_H	MailBox-to-queue mapping register	Section 3.10.1.1
063Ch	RXU_MAP47_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0640h	RXU_MAP48_L	MailBox-to-queue mapping register	Section 3.10.1.1
0644h	RXU_MAP48_H	MailBox-to-queue mapping register	Section 3.10.1.1
0648h	RXU_MAP48_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
064Ch	RXU_MAP49_L	MailBox-to-queue mapping register	Section 3.10.1.1
0650h	RXU_MAP49_H	MailBox-to-queue mapping register	Section 3.10.1.1
0654h	RXU_MAP49_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0658h	RXU_MAP50_L	MailBox-to-queue mapping register	Section 3.10.1.1
065Ch	RXU_MAP50_H	MailBox-to-queue mapping register	Section 3.10.1.1
0660h	RXU_MAP50_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0664h	RXU_MAP51_L	MailBox-to-queue mapping register	Section 3.10.1.1
0668h	RXU_MAP51_H	MailBox-to-queue mapping register	Section 3.10.1.1
066Ch	RXU_MAP51_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0670h	RXU_MAP52_L	MailBox-to-queue mapping register	Section 3.10.1.1
0674h	RXU_MAP52_H	MailBox-to-queue mapping register	Section 3.10.1.1
0678h	RXU_MAP52_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
067Ch	RXU_MAP53_L	MailBox-to-queue mapping register	Section 3.10.1.1
0680h	RXU_MAP53_H	MailBox-to-queue mapping register	Section 3.10.1.1
0684h	RXU_MAP53_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0688h	RXU_MAP54_L	MailBox-to-queue mapping register	Section 3.10.1.1
068Ch	RXU_MAP54_H	MailBox-to-queue mapping register	Section 3.10.1.1
0690h	RXU_MAP54_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0694h	RXU_MAP55_L	MailBox-to-queue mapping register	Section 3.10.1.1
0698h	RXU_MAP55_H	MailBox-to-queue mapping register	Section 3.10.1.1
069Ch	RXU_MAP55_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06A0h	RXU_MAP56_L	MailBox-to-queue mapping register	Section 3.10.1.1
06A4h	RXU_MAP56_H	MailBox-to-queue mapping register	Section 3.10.1.1
06A8h	RXU_MAP56_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06ACh	RXU_MAP57_L	MailBox-to-queue mapping register	Section 3.10.1.1
06B0h	RXU_MAP57_H	MailBox-to-queue mapping register	Section 3.10.1.1
06B4h	RXU_MAP57_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06B8h	RXU_MAP58_L	MailBox-to-queue mapping register	Section 3.10.1.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
06BCh	RXU_MAP58_H	MailBox-to-queue mapping register	Section 3.10.1.1
06C0h	RXU_MAP58_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06C4h	RXU_MAP59_L	MailBox-to-queue mapping register	Section 3.10.1.1
06C8h	RXU_MAP59_H	MailBox-to-queue mapping register	Section 3.10.1.1
06CCh	RXU_MAP59_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06D0h	RXU_MAP60_L	MailBox-to-queue mapping register	Section 3.10.1.1
06D4h	RXU_MAP60_H	MailBox-to-queue mapping register	Section 3.10.1.1
06D8h	RXU_MAP60_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06DCh	RXU_MAP61_L	MailBox-to-queue mapping register	Section 3.10.1.1
06E0h	RXU_MAP61_H	MailBox-to-queue mapping register	Section 3.10.1.1
06E4h	RXU_MAP61_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06E8h	RXU_MAP62_L	MailBox-to-queue mapping register	Section 3.10.1.1
06ECh	RXU_MAP62_H	MailBox-to-queue mapping register	Section 3.10.1.1
06F0h	RXU_MAP62_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
06F4h	RXU_MAP63_L	MailBox-to-queue mapping register	Section 3.10.1.1
06F8h	RXU_MAP63_H	MailBox-to-queue mapping register	Section 3.10.1.1
06FCh	RXU_MAP63_QID	CPPI Queue ID for this mapping reg	Section 3.10.3.1
0700h	RXU_TYPE9_MAP00_0	Type 9 Mapping Register 0	Section 3.10.2.1
0704h	RXU_TYPE9_MAP00_1	Type 9 Mapping Register 1	Section 3.10.2.2

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0708h	RXU_TYPE9_MAP00_2	Type 9 Mapping Register 2	Section 3.10.2.3
070Ch	RXU_TYPE9_MAP01_0	Type 9 Mapping Register 0	Section 3.10.2.1
0710h	RXU_TYPE9_MAP01_1	Type 9 Mapping Register 1	Section 3.10.2.2
0714h	RXU_TYPE9_MAP01_2	Type 9 Mapping Register 2	Section 3.10.2.3
0718h	RXU_TYPE9_MAP02_0	Type 9 Mapping Register 0	Section 3.10.2.1
071Ch	RXU_TYPE9_MAP02_1	Type 9 Mapping Register 1	Section 3.10.2.2
0720h	RXU_TYPE9_MAP02_2	Type 9 Mapping Register 2	Section 3.10.2.3
0724h	RXU_TYPE9_MAP03_0	Type 9 Mapping Register 0	Section 3.10.2.1
0728h	RXU_TYPE9_MAP03_1	Type 9 Mapping Register 1	Section 3.10.2.2
072Ch	RXU_TYPE9_MAP03_2	Type 9 Mapping Register 2	Section 3.10.2.3
0730h	RXU_TYPE9_MAP04_0	Type 9 Mapping Register 0	Section 3.10.2.1
0734h	RXU_TYPE9_MAP04_1	Type 9 Mapping Register 1	Section 3.10.2.2
0738h	RXU_TYPE9_MAP04_2	Type 9 Mapping Register 2	Section 3.10.2.3
073Ch	RXU_TYPE9_MAP05_0	Type 9 Mapping Register 0	Section 3.10.2.1
0740h	RXU_TYPE9_MAP05_1	Type 9 Mapping Register 1	Section 3.10.2.2
0744h	RXU_TYPE9_MAP05_2	Type 9 Mapping Register 2	Section 3.10.2.3
0748h	RXU_TYPE9_MAP06_0	Type 9 Mapping Register 0	Section 3.10.2.1
074Ch	RXU_TYPE9_MAP06_1	Type 9 Mapping Register 1	Section 3.10.2.2
0750h	RXU_TYPE9_MAP06_2	Type 9 Mapping Register 2	Section 3.10.2.3

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0754h	RXU_TYPE9_MAP07_0	Type 9 Mapping Register 0	Section 3.10.2.1
0758h	RXU_TYPE9_MAP07_1	Type 9 Mapping Register 1	Section 3.10.2.2
075Ch	RXU_TYPE9_MAP07_2	Type 9 Mapping Register 2	Section 3.10.2.3
0760h	RXU_TYPE9_MAP08_0	Type 9 Mapping Register 0	Section 3.10.2.1
0764h	RXU_TYPE9_MAP08_1	Type 9 Mapping Register 1	Section 3.10.2.2
0768h	RXU_TYPE9_MAP08_2	Type 9 Mapping Register 2	Section 3.10.2.3
076Ch	RXU_TYPE9_MAP09_0	Type 9 Mapping Register 0	Section 3.10.2.1
0770h	RXU_TYPE9_MAP09_1	Type 9 Mapping Register 1	Section 3.10.2.2
0774h	RXU_TYPE9_MAP09_2	Type 9 Mapping Register 2	Section 3.10.2.3
0778h	RXU_TYPE9_MAP10_0	Type 9 Mapping Register 0	Section 3.10.2.1
077Ch	RXU_TYPE9_MAP10_1	Type 9 Mapping Register 1	Section 3.10.2.2
0780h	RXU_TYPE9_MAP10_2	Type 9 Mapping Register 2	Section 3.10.2.3
0784h	RXU_TYPE9_MAP11_0	Type 9 Mapping Register 0	Section 3.10.2.1
0788h	RXU_TYPE9_MAP11_1	Type 9 Mapping Register 1	Section 3.10.2.2
078Ch	RXU_TYPE9_MAP11_2	Type 9 Mapping Register 2	Section 3.10.2.3
0790h	RXU_TYPE9_MAP12_0	Type 9 Mapping Register 0	Section 3.10.2.1
0794h	RXU_TYPE9_MAP12_1	Type 9 Mapping Register 1	Section 3.10.2.2
0798h	RXU_TYPE9_MAP12_2	Type 9 Mapping Register 2	Section 3.10.2.3
079Ch	RXU_TYPE9_MAP13_0	Type 9 Mapping Register 0	Section 3.10.2.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
07A0h	RXU_TYPE9_MAP13_1	Type 9 Mapping Register 1	Section 3.10.2.2
07A4h	RXU_TYPE9_MAP13_2	Type 9 Mapping Register 2	Section 3.10.2.3
07A8h	RXU_TYPE9_MAP14_0	Type 9 Mapping Register 0	Section 3.10.2.1
07ACh	RXU_TYPE9_MAP14_1	Type 9 Mapping Register 1	Section 3.10.2.2
07B0h	RXU_TYPE9_MAP14_2	Type 9 Mapping Register 2	Section 3.10.2.3
07B4h	RXU_TYPE9_MAP15_0	Type 9 Mapping Register 0	Section 3.10.2.1
07B8h	RXU_TYPE9_MAP15_1	Type 9 Mapping Register 1	Section 3.10.2.2
07BCh	RXU_TYPE9_MAP15_2	Type 9 Mapping Register 2	Section 3.10.2.3
07C0h	RXU_TYPE9_MAP16_0	Type 9 Mapping Register 0	Section 3.10.2.1
07C4h	RXU_TYPE9_MAP16_1	Type 9 Mapping Register 1	Section 3.10.2.2
07C8h	RXU_TYPE9_MAP16_2	Type 9 Mapping Register 2	Section 3.10.2.3
07CCh	RXU_TYPE9_MAP17_0	Type 9 Mapping Register 0	Section 3.10.2.1
07D0h	RXU_TYPE9_MAP17_1	Type 9 Mapping Register 1	Section 3.10.2.2
07D4h	RXU_TYPE9_MAP17_2	Type 9 Mapping Register 2	Section 3.10.2.3
07D8h	RXU_TYPE9_MAP18_0	Type 9 Mapping Register 0	Section 3.10.2.1
07DCh	RXU_TYPE9_MAP18_1	Type 9 Mapping Register 1	Section 3.10.2.2
07E0h	RXU_TYPE9_MAP18_2	Type 9 Mapping Register 2	Section 3.10.2.3
07E4h	RXU_TYPE9_MAP19_0	Type 9 Mapping Register 0	Section 3.10.2.1
07E8h	RXU_TYPE9_MAP19_1	Type 9 Mapping Register 1	Section 3.10.2.2

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
07ECh	RXU_TYPE9_MAP19_2	Type 9 Mapping Register 2	Section 3.10.2.3
07F0h	RXU_TYPE9_MAP20_0	Type 9 Mapping Register 0	Section 3.10.2.1
07F4h	RXU_TYPE9_MAP20_1	Type 9 Mapping Register 1	Section 3.10.2.2
07F8h	RXU_TYPE9_MAP20_2	Type 9 Mapping Register 2	Section 3.10.2.3
07FCh	RXU_TYPE9_MAP21_0	Type 9 Mapping Register 0	Section 3.10.2.1
0800h	RXU_TYPE9_MAP21_1	Type 9 Mapping Register 1	Section 3.10.2.2
0804h	RXU_TYPE9_MAP21_2	Type 9 Mapping Register 2	Section 3.10.2.3
0808h	RXU_TYPE9_MAP22_0	Type 9 Mapping Register 0	Section 3.10.2.1
080Ch	RXU_TYPE9_MAP22_1	Type 9 Mapping Register 1	Section 3.10.2.2
0810h	RXU_TYPE9_MAP22_2	Type 9 Mapping Register 2	Section 3.10.2.3
0814h	RXU_TYPE9_MAP23_0	Type 9 Mapping Register 0	Section 3.10.2.1
0818h	RXU_TYPE9_MAP23_1	Type 9 Mapping Register 1	Section 3.10.2.2
081Ch	RXU_TYPE9_MAP23_2	Type 9 Mapping Register 2	Section 3.10.2.3
0820h	RXU_TYPE9_MAP24_0	Type 9 Mapping Register 0	Section 3.10.2.1
0824h	RXU_TYPE9_MAP24_1	Type 9 Mapping Register 1	Section 3.10.2.2
0828h	RXU_TYPE9_MAP24_2	Type 9 Mapping Register 2	Section 3.10.2.3
082Ch	RXU_TYPE9_MAP25_0	Type 9 Mapping Register 0	Section 3.10.2.1
0830h	RXU_TYPE9_MAP25_1	Type 9 Mapping Register 1	Section 3.10.2.2
0834h	RXU_TYPE9_MAP25_2	Type 9 Mapping Register 2	Section 3.10.2.3

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0838h	RXU_TYPE9_MAP26_0	Type 9 Mapping Register 0	Section 3.10.2.1
083Ch	RXU_TYPE9_MAP26_1	Type 9 Mapping Register 1	Section 3.10.2.2
0840h	RXU_TYPE9_MAP26_2	Type 9 Mapping Register 2	Section 3.10.2.3
0844h	RXU_TYPE9_MAP27_0	Type 9 Mapping Register 0	Section 3.10.2.1
0848h	RXU_TYPE9_MAP27_1	Type 9 Mapping Register 1	Section 3.10.2.2
084Ch	RXU_TYPE9_MAP27_2	Type 9 Mapping Register 2	Section 3.10.2.3
0850h	RXU_TYPE9_MAP28_0	Type 9 Mapping Register 0	Section 3.10.2.1
0854h	RXU_TYPE9_MAP28_1	Type 9 Mapping Register 1	Section 3.10.2.2
0858h	RXU_TYPE9_MAP28_2	Type 9 Mapping Register 2	Section 3.10.2.3
085Ch	RXU_TYPE9_MAP29_0	Type 9 Mapping Register 0	Section 3.10.2.1
0860h	RXU_TYPE9_MAP29_1	Type 9 Mapping Register 1	Section 3.10.2.2
0864h	RXU_TYPE9_MAP29_2	Type 9 Mapping Register 2	Section 3.10.2.3
0868h	RXU_TYPE9_MAP30_0	Type 9 Mapping Register 0	Section 3.10.2.1
086Ch	RXU_TYPE9_MAP30_1	Type 9 Mapping Register 1	Section 3.10.2.2
0870h	RXU_TYPE9_MAP30_2	Type 9 Mapping Register 2	Section 3.10.2.3
0874h	RXU_TYPE9_MAP31_0	Type 9 Mapping Register 0	Section 3.10.2.1
0878h	RXU_TYPE9_MAP31_1	Type 9 Mapping Register 1	Section 3.10.2.2
087Ch	RXU_TYPE9_MAP31_2	Type 9 Mapping Register 2	Section 3.10.2.3
0880h	RXU_TYPE9_MAP32_0	Type 9 Mapping Register 0	Section 3.10.2.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0884h	RXU_TYPE9_MAP32_1	Type 9 Mapping Register 1	Section 3.10.2.2
0888h	RXU_TYPE9_MAP32_2	Type 9 Mapping Register 2	Section 3.10.2.3
088Ch	RXU_TYPE9_MAP33_0	Type 9 Mapping Register 0	Section 3.10.2.1
0890h	RXU_TYPE9_MAP33_1	Type 9 Mapping Register 1	Section 3.10.2.2
0894h	RXU_TYPE9_MAP33_2	Type 9 Mapping Register 2	Section 3.10.2.3
0898h	RXU_TYPE9_MAP34_0	Type 9 Mapping Register 0	Section 3.10.2.1
089Ch	RXU_TYPE9_MAP34_1	Type 9 Mapping Register 1	Section 3.10.2.2
08A0h	RXU_TYPE9_MAP34_2	Type 9 Mapping Register 2	Section 3.10.2.3
08A4h	RXU_TYPE9_MAP35_0	Type 9 Mapping Register 0	Section 3.10.2.1
08A8h	RXU_TYPE9_MAP35_1	Type 9 Mapping Register 1	Section 3.10.2.2
08ACh	RXU_TYPE9_MAP35_2	Type 9 Mapping Register 2	Section 3.10.2.3
08B0h	RXU_TYPE9_MAP36_0	Type 9 Mapping Register 0	Section 3.10.2.1
08B4h	RXU_TYPE9_MAP36_1	Type 9 Mapping Register 1	Section 3.10.2.2
08B8h	RXU_TYPE9_MAP36_2	Type 9 Mapping Register 2	Section 3.10.2.3
08BCh	RXU_TYPE9_MAP37_0	Type 9 Mapping Register 0	Section 3.10.2.1
08C0h	RXU_TYPE9_MAP37_1	Type 9 Mapping Register 1	Section 3.10.2.2
08C4h	RXU_TYPE9_MAP37_2	Type 9 Mapping Register 2	Section 3.10.2.3
08C8h	RXU_TYPE9_MAP38_0	Type 9 Mapping Register 0	Section 3.10.2.1
08CCh	RXU_TYPE9_MAP38_1	Type 9 Mapping Register 1	Section 3.10.2.2

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
08D0h	RXU_TYPE9_MAP38_2	Type 9 Mapping Register 2	Section 3.10.2.3
08D4h	RXU_TYPE9_MAP39_0	Type 9 Mapping Register 0	Section 3.10.2.1
08D8h	RXU_TYPE9_MAP39_1	Type 9 Mapping Register 1	Section 3.10.2.2
08DCh	RXU_TYPE9_MAP39_2	Type 9 Mapping Register 2	Section 3.10.2.3
08E0h	RXU_TYPE9_MAP40_0	Type 9 Mapping Register 0	Section 3.10.2.1
08E4h	RXU_TYPE9_MAP40_1	Type 9 Mapping Register 1	Section 3.10.2.2
08E8h	RXU_TYPE9_MAP40_2	Type 9 Mapping Register 2	Section 3.10.2.3
08ECh	RXU_TYPE9_MAP41_0	Type 9 Mapping Register 0	Section 3.10.2.1
08F0h	RXU_TYPE9_MAP41_1	Type 9 Mapping Register 1	Section 3.10.2.2
08F4h	RXU_TYPE9_MAP41_2	Type 9 Mapping Register 2	Section 3.10.2.3
08F8h	RXU_TYPE9_MAP42_0	Type 9 Mapping Register 0	Section 3.10.2.1
08FCh	RXU_TYPE9_MAP42_1	Type 9 Mapping Register 1	Section 3.10.2.2
0900h	RXU_TYPE9_MAP42_2	Type 9 Mapping Register 2	Section 3.10.2.3
0904h	RXU_TYPE9_MAP43_0	Type 9 Mapping Register 0	Section 3.10.2.1
0908h	RXU_TYPE9_MAP43_1	Type 9 Mapping Register 1	Section 3.10.2.2
090Ch	RXU_TYPE9_MAP43_2	Type 9 Mapping Register 2	Section 3.10.2.3
0910h	RXU_TYPE9_MAP44_0	Type 9 Mapping Register 0	Section 3.10.2.1
0914h	RXU_TYPE9_MAP44_1	Type 9 Mapping Register 1	Section 3.10.2.2
0918h	RXU_TYPE9_MAP44_2	Type 9 Mapping Register 2	Section 3.10.2.3

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
091Ch	RXU_TYPE9_MAP45_0	Type 9 Mapping Register 0	Section 3.10.2.1
0920h	RXU_TYPE9_MAP45_1	Type 9 Mapping Register 1	Section 3.10.2.2
0924h	RXU_TYPE9_MAP45_2	Type 9 Mapping Register 2	Section 3.10.2.3
0928h	RXU_TYPE9_MAP46_0	Type 9 Mapping Register 0	Section 3.10.2.1
092Ch	RXU_TYPE9_MAP46_1	Type 9 Mapping Register 1	Section 3.10.2.2
0930h	RXU_TYPE9_MAP46_2	Type 9 Mapping Register 2	Section 3.10.2.3
0934h	RXU_TYPE9_MAP47_0	Type 9 Mapping Register 0	Section 3.10.2.1
0938h	RXU_TYPE9_MAP47_1	Type 9 Mapping Register 1	Section 3.10.2.2
093Ch	RXU_TYPE9_MAP47_2	Type 9 Mapping Register 2	Section 3.10.2.3
0940h	RXU_TYPE9_MAP48_0	Type 9 Mapping Register 0	Section 3.10.2.1
0944h	RXU_TYPE9_MAP48_1	Type 9 Mapping Register 1	Section 3.10.2.2
0948h	RXU_TYPE9_MAP48_2	Type 9 Mapping Register 2	Section 3.10.2.3
094Ch	RXU_TYPE9_MAP49_0	Type 9 Mapping Register 0	Section 3.10.2.1
0950h	RXU_TYPE9_MAP49_1	Type 9 Mapping Register 1	Section 3.10.2.2
0954h	RXU_TYPE9_MAP49_2	Type 9 Mapping Register 2	Section 3.10.2.3
0958h	RXU_TYPE9_MAP50_0	Type 9 Mapping Register 0	Section 3.10.2.1
095Ch	RXU_TYPE9_MAP50_1	Type 9 Mapping Register 1	Section 3.10.2.2
0960h	RXU_TYPE9_MAP50_2	Type 9 Mapping Register 2	Section 3.10.2.3
0964h	RXU_TYPE9_MAP51_0	Type 9 Mapping Register 0	Section 3.10.2.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0968h	RXU_TYPE9_MAP51_1	Type 9 Mapping Register 1	Section 3.10.2.2
096Ch	RXU_TYPE9_MAP51_2	Type 9 Mapping Register 2	Section 3.10.2.3
0970h	RXU_TYPE9_MAP52_0	Type 9 Mapping Register 0	Section 3.10.2.1
0974h	RXU_TYPE9_MAP52_1	Type 9 Mapping Register 1	Section 3.10.2.2
0978h	RXU_TYPE9_MAP52_2	Type 9 Mapping Register 2	Section 3.10.2.3
097Ch	RXU_TYPE9_MAP53_0	Type 9 Mapping Register 0	Section 3.10.2.1
0980h	RXU_TYPE9_MAP53_1	Type 9 Mapping Register 1	Section 3.10.2.2
0984h	RXU_TYPE9_MAP53_2	Type 9 Mapping Register 2	Section 3.10.2.3
0988h	RXU_TYPE9_MAP54_0	Type 9 Mapping Register 0	Section 3.10.2.1
098Ch	RXU_TYPE9_MAP54_1	Type 9 Mapping Register 1	Section 3.10.2.2
0990h	RXU_TYPE9_MAP54_2	Type 9 Mapping Register 2	Section 3.10.2.3
0994h	RXU_TYPE9_MAP55_0	Type 9 Mapping Register 0	Section 3.10.2.1
0998h	RXU_TYPE9_MAP55_1	Type 9 Mapping Register 1	Section 3.10.2.2
099Ch	RXU_TYPE9_MAP55_2	Type 9 Mapping Register 2	Section 3.10.2.3
09A0h	RXU_TYPE9_MAP56_0	Type 9 Mapping Register 0	Section 3.10.2.1
09A4h	RXU_TYPE9_MAP56_1	Type 9 Mapping Register 1	Section 3.10.2.2
09A8h	RXU_TYPE9_MAP56_2	Type 9 Mapping Register 2	Section 3.10.2.3
09ACh	RXU_TYPE9_MAP57_0	Type 9 Mapping Register 0	Section 3.10.2.1
09B0h	RXU_TYPE9_MAP57_1	Type 9 Mapping Register 1	Section 3.10.2.2

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
09B4h	RXU_TYPE9_MAP57_2	Type 9 Mapping Register 2	Section 3.10.2.3
09B8h	RXU_TYPE9_MAP58_0	Type 9 Mapping Register 0	Section 3.10.2.1
09BCh	RXU_TYPE9_MAP58_1	Type 9 Mapping Register 1	Section 3.10.2.2
09C0h	RXU_TYPE9_MAP58_2	Type 9 Mapping Register 2	Section 3.10.2.3
09C4h	RXU_TYPE9_MAP59_0	Type 9 Mapping Register 0	Section 3.10.2.1
09C8h	RXU_TYPE9_MAP59_1	Type 9 Mapping Register 1	Section 3.10.2.2
09CCh	RXU_TYPE9_MAP59_2	Type 9 Mapping Register 2	Section 3.10.2.3
09D0h	RXU_TYPE9_MAP60_0	Type 9 Mapping Register 0	Section 3.10.2.1
09D4h	RXU_TYPE9_MAP60_1	Type 9 Mapping Register 1	Section 3.10.2.2
09D8h	RXU_TYPE9_MAP60_2	Type 9 Mapping Register 2	Section 3.10.2.3
09DCh	RXU_TYPE9_MAP61_0	Type 9 Mapping Register 0	Section 3.10.2.1
09E0h	RXU_TYPE9_MAP61_1	Type 9 Mapping Register 1	Section 3.10.2.2
09E4h	RXU_TYPE9_MAP61_2	Type 9 Mapping Register 2	Section 3.10.2.3
09E8h	RXU_TYPE9_MAP62_0	Type 9 Mapping Register 0	Section 3.10.2.1
09ECh	RXU_TYPE9_MAP62_1	Type 9 Mapping Register 1	Section 3.10.2.2
09F0h	RXU_TYPE9_MAP62_2	Type 9 Mapping Register 2	Section 3.10.2.3
09F4h	RXU_TYPE9_MAP63_0	Type 9 Mapping Register 0	Section 3.10.2.1
09F8h	RXU_TYPE9_MAP63_1	Type 9 Mapping Register 1	Section 3.10.2.2
09FCh	RXU_TYPE9_MAP63_2	Type 9 Mapping Register 2	Section 3.10.2.3
LSU/MAU REGISTERS			

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0D00h	LSU1_Reg0	LSU1 Control Reg0	Section 3.11.1
0D04h	LSU1_Reg1	LSU1 Control Reg1	Section 3.11.2
0D08h	LSU1_Reg2	LSU1 Control Reg2	Section 3.11.3
0D0Ch	LSU1_Reg3	LSU1 Control Reg3	Section 3.11.4
0D10h	LSU1_Reg4	LSU1 Control Reg4	Section 3.11.5
0D14h	LSU1_Reg5	LSU1 Command Reg5	Section 3.11.6
0D18h	LSU1_Reg6	LSU1 Status Reg6	Section 3.11.7
0D1Ch	LSU2_Reg0	LSU2 Control Reg0	Section 3.11.1
0D20h	LSU2_Reg1	LSU2 Control Reg1	Section 3.11.2
0D24h	LSU2_Reg2	LSU2 Control Reg2	Section 3.11.3
0D28h	LSU2_Reg3	LSU2 Control Reg3	Section 3.11.4
0D2Ch	LSU2_Reg4	LSU2 Control Reg4	Section 3.11.5
0D30h	LSU2_Reg5	LSU2 Command Reg5	Section 3.11.6
0D34h	LSU2_Reg6	LSU2 Status Reg6	Section 3.11.7
0D38h	LSU3_Reg0	LSU3 Control Reg0	Section 3.11.1
0D3Ch	LSU3_Reg1	LSU3 Control Reg1	Section 3.11.2
0D40h	LSU3_Reg2	LSU3 Control Reg2	Section 3.11.3
0D44h	LSU3_Reg3	LSU3 Control Reg3	Section 3.11.4
0D48h	LSU3_Reg4	LSU3 Control Reg4	Section 3.11.5
0D4Ch	LSU3_Reg5	LSU3 Command Reg5	Section 3.11.6
0D50h	LSU3_Reg6	LSU3 Status Reg6	Section 3.11.7
0D54h	LSU4_Reg0	LSU4 Control Reg0	Section 3.11.1
0D58h	LSU4_Reg1	LSU4 Control Reg1	Section 3.11.2
0D5Ch	LSU4_Reg2	LSU4 Control Reg2	Section 3.11.3
0D60h	LSU4_Reg3	LSU4 Control Reg3	Section 3.11.4
0D64h	LSU4_Reg4	LSU4 Control Reg4	Section 3.11.5
0D68h	LSU4_Reg5	LSU4 Command Reg5	Section 3.11.6

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0D6Ch	LSU4_Reg6	LSU4 Status Reg6	Section 3.11.7
0D70h	LSU5_Reg0	LSU5 Control Reg0	Section 3.11.1
0D74h	LSU5_Reg1	LSU5 Control Reg1	Section 3.11.2
0D78h	LSU5_Reg2	LSU5 Control Reg2	Section 3.11.3
0D7Ch	LSU5_Reg3	LSU5 Control Reg3	Section 3.11.4
0D80h	LSU5_Reg4	LSU5 Control Reg4	Section 3.11.5
0D84h	LSU5_Reg5	LSU5 Command Reg5	Section 3.11.6
0D88h	LSU5_Reg6	LSU5 Status Reg6	Section 3.11.7
0D8Ch	LSU6_Reg0	LSU6 Control Reg0	Section 3.11.1
0D90h	LSU6_Reg1	LSU6 Control Reg1	Section 3.11.2
0D94h	LSU6_Reg2	LSU6 Control Reg2	Section 3.11.3
0D98h	LSU6_Reg3	LSU6 Control Reg3	Section 3.11.4
0D9Ch	LSU6_Reg4	LSU6 Control Reg4	Section 3.11.5
0DA0h	LSU6_Reg5	LSU6 Command Reg5	Section 3.11.6
0DA4h	LSU6_Reg6	LS62 Status Reg6	Section 3.11.7
0DA8h	LSU7_Reg0	LSU7 Control Reg0	Section 3.11.1
0DACh	LSU7_Reg1	LSU7 Control Reg1	Section 3.11.2
0DB0h	LSU7_Reg2	LSU7 Control Reg2	Section 3.11.3
0DB4h	LSU7_Reg3	LSU7 Control Reg3	Section 3.11.4
0DB8h	LSU7_Reg4	LSU7 Control Reg4	Section 3.11.5
0DBCh	LSU7_Reg5	LSU7 Command Reg5	Section 3.11.6
0DC0h	LSU7_Reg6	LSU7 Status Reg6	Section 3.11.7
0DC4h	LSU8_Reg0	LSU8 Control Reg0	Section 3.11.1
0DC8h	LSU8_Reg1	LSU8 Control Reg1	Section 3.11.2
0DCCh	LSU8_Reg2	LSU8 Control Reg2	Section 3.11.3
0DD0h	LSU8_Reg3	LSU8 Control Reg3	Section 3.11.4
0DD4h	LSU8_Reg4	LSU8 Control Reg4	Section 3.11.5

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0DD8h	LSU8_Reg5	LSU8 Command Reg5	Section 3.11.6
0DDCh	LSU8_Reg6	LSU8 Status Reg6	Section 3.11.7
0DE0h	LSU_SETUP_REG0	Setup for LSU shadow registers	Section 3.11.8
0DE4h	LSU_SETUP_REG1	Setup for LSUs	Section 3.11.9
0DE8h	LSU_STAT_REG0	Status for shadow register	Section 3.11.10
0DECh	LSU_STAT_REG1	Status for shadow register	Section 3.11.11
0DF0h	LSU_STAT_REG2	Status for shadow register	Section 3.11.12
0DF4h	LSU_STAT_REG3	Status for shadow register	Section 3.11.13
0DF8h	LSU_STAT_REG4	Status for shadow register	Section 3.11.14
0DFCh	LSU_STAT_REG5	Status for shadow register	Section 3.11.15
0E00h	LSU_FLOW_MASKS0	LSU1,2 Congestion Control Flow Mask	Section 3.11.16
0E04h	LSU_FLOW_MASKS1	LSU3,4 Congestion Control Flow Mask	Section 3.11.17
0E08h	LSU_FLOW_MASKS2	LSU5,6 Congestion Control Flow Mask	Section 3.11.18
0E0Ch	LSU_FLOW_MASKS3	LSU7,8 Congestion Control Flow Mask	Section 3.11.19
0E4Ch	SUPRVSR_ID	DeviceID allowed Supervisor permissions for MAU	Section 3.11.20
FLOW CONTROL REGISTERS			
0E50h	FLOW_CNTL0	Flow Control Table Entry Register0	Section 3.12.1
0E54h	FLOW_CNTL1	Flow Control Table Entry Register1	Section 3.12.1
0E58h	FLOW_CNTL2	Flow Control Table Entry Register2	Section 3.12.1
0E5Ch	FLOW_CNTL3	Flow Control Table Entry Register3	Section 3.12.1
0E60h	FLOW_CNTL4	Flow Control Table Entry Register4	Section 3.12.1
0E64h	FLOW_CNTL5	Flow Control Table Entry Register5	Section 3.12.1
0E68h	FLOW_CNTL6	Flow Control Table Entry Register6	Section 3.12.1
0E6Ch	FLOW_CNTL7	Flow Control Table Entry Register7	Section 3.12.1
0E70h	FLOW_CNTL8	Flow Control Table Entry Register8	Section 3.12.1
0E74h	FLOW_CNTL9	Flow Control Table Entry Register9	Section 3.12.1
0E78h	FLOW_CNTL10	Flow Control Table Entry Register10	Section 3.12.1
0E7Ch	FLOW_CNTL11	Flow Control Table Entry Register11	Section 3.12.1

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
0E80h	FLOW_CNTL12	Flow Control Table Entry Register12	Section 3.12.1
0E84h	FLOW_CNTL13	Flow Control Table Entry Register13	Section 3.12.1
0E88h	FLOW_CNTL14	Flow Control Table Entry Register14	Section 3.12.1
0E8Ch	FLOW_CNTL15	Flow Control Table Entry Register15	Section 3.12.1
TXU REGISTERS			
0EB0h	TX_CPPI_FLOW_MASKS0	TX CPPI Supported Flow Mask	Section 3.13.1
0EB4h	TX_CPPI_FLOW_MASKS1	TX CPPI Supported Flow Mask	Section 3.13.1
0EB8h	TX_CPPI_FLOW_MASKS2	TX CPPI Supported Flow Mask	Section 3.13.1
0EBCh	TX_CPPI_FLOW_MASKS3	TX CPPI Supported Flow Mask	Section 3.13.1
0EC0h	TX_CPPI_FLOW_MASKS4	TX CPPI Supported Flow Mask	Section 3.13.1
0EC4h	TX_CPPI_FLOW_MASKS5	TX CPPI Supported Flow Mask	Section 3.13.1
0EC8h	TX_CPPI_FLOW_MASKS6	TX CPPI Supported Flow Mask	Section 3.13.1
0ECCh	TX_CPPI_FLOW_MASKS7	TX CPPI Supported Flow Mask	Section 3.13.1
0ED0h	TX_QUEUE_SCH_INFO1	TX Queue Priority and Port Info	Section 3.13.2
0ED4h	TX_QUEUE_SCH_INFO2	TX Queue Priority and Port Info	Section 3.13.2
0ED8h	TX_QUEUE_SCH_INFO3	TX Queue Priority and Port Info	Section 3.13.2
0EDCh	TX_QUEUE_SCH_INFO4	TX Queue Priority and Port Info	Section 3.13.2
0EE0h	Garbage_Coll_QID0	Queue ID used for garbage collection of TX descriptors under error	Section 3.13.3
0EE4h	Garbage_Coll_QID1	Queue ID used for garbage collection of TX descriptors under error	Section 3.13.4
0EE8h	Garbage_Coll_QID2	Queue ID used for garbage collection of TX descriptors under error	Section 3.13.5
PKTDMA CONTROL REGISTER REGION			
1000h	REVISION_REG	Revision Register	See Section 3.14
1004h	PERF_CTRL_REG	Performance Control Register	
1008h	EMU_CTRL_REG	Emulation Control Register	
100Ch	PRI_CTRL_REG	Priority Control Register	
PKTDMA TX DMA CHANNEL CONTROL/STATUS REGISTERS			

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1400h	TCHAN_GCFG_REG_A0	Tx Channel 0 Global Config Register A	See Section 3.14
1404h	TCHAN_GCFG_REG_B0	Tx Channel 0 Global Config Register B	
1420h	TCHAN_GCFG_REG_A1	Tx Channel 1 Global Config Register A	
1424h	TCHAN_GCFG_REG_B1	Tx Channel 1 Global Config Register B	
1440h	TCHAN_GCFG_REG_A2	Tx Channel 2 Global Config Register A	
1444h	TCHAN_GCFG_REG_B2	Tx Channel 2 Global Config Register B	
1460h	TCHAN_GCFG_REG_A3	Tx Channel 3 Global Config Register A	
1464h	TCHAN_GCFG_REG_B3	Tx Channel 3 Global Config Register B	
1480h	TCHAN_GCFG_REG_A4	Tx Channel 4 Global Config Register A	
1484h	TCHAN_GCFG_REG_B4	Tx Channel 4 Global Config Register B	
14A0h	TCHAN_GCFG_REG_A5	Tx Channel 5 Global Config Register A	
14A4h	TCHAN_GCFG_REG_B5	Tx Channel 5 Global Config Register B	
14C0h	TCHAN_GCFG_REG_A6	Tx Channel 6 Global Config Register A	
14C4h	TCHAN_GCFG_REG_B6	Tx Channel 6 Global Config Register B	See Section 3.14
14E0h	TCHAN_GCFG_REG_A7	Tx Channel 7 Global Config Register A	
14E4h	TCHAN_GCFG_REG_B7	Tx Channel 7 Global Config Register B	
1500h	TCHAN_GCFG_REG_A8	Tx Channel 8 Global Config Register A	
1504h	TCHAN_GCFG_REG_B8	Tx Channel 8 Global Config Register B	
1520h	TCHAN_GCFG_REG_A9	Tx Channel 9 Global Config Register A	
1524h	TCHAN_GCFG_REG_B9	Tx Channel 9 Global Config Register B	
1540h	TCHAN_GCFG_REG_A10	Tx Channel 10 Global Config Register A	
1544h	TCHAN_GCFG_REG_B10	Tx Channel 10 Global Config Register B	
1560h	TCHAN_GCFG_REG_A11	Tx Channel 11 Global Config Register A	
1564h	TCHAN_GCFG_REG_B11	Tx Channel 11 Global Config Register B	
1580h	TCHAN_GCFG_REG_A12	Tx Channel 12 Global Config Register A	
1584h	TCHAN_GCFG_REG_B12	Tx Channel 12 Global Config Register B	
15A0h	TCHAN_GCFG_REG_A13	Tx Channel 13 Global Config Register A	
15A4h	TCHAN_GCFG_REG_B13	Tx Channel 13 Global Config Register B	
15C0h	TCHAN_GCFG_REG_A14	Tx Channel 14 Global Config Register A	
15C4h	TCHAN_GCFG_REG_B14	Tx Channel 14 Global Config Register B	
15E0h	TCHAN_GCFG_REG_A15	Tx Channel 15 Global Config Register A	
15E4h	TCHAN_GCFG_REG_B15	Tx Channel 15 Global Config Register B	
PKTDMA RX DMA CHANNEL CONTROL/STATUS REGISTERS			

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1800h	RCHAN_GCFG_REG_A0	Rx Channel 0 Global Config Register A	See Section 3.14
1820h	RCHAN_GCFG_REG_A1	Rx Channel 1 Global Config Register A	
1840h	RCHAN_GCFG_REG_A2	Rx Channel 2 Global Config Register A	
1860h	RCHAN_GCFG_REG_A3	Rx Channel 3 Global Config Register A	
1880h	RCHAN_GCFG_REG_A4	Rx Channel 4 Global Config Register A	
18A0h	RCHAN_GCFG_REG_A5	Rx Channel 5 Global Config Register A	
18C0h	RCHAN_GCFG_REG_A6	Rx Channel 6 Global Config Register A	
18E0h	RCHAN_GCFG_REG_A7	Rx Channel 7 Global Config Register A	
1900h	RCHAN_GCFG_REG_A8	Rx Channel 8 Global Config Register A	
1920h	RCHAN_GCFG_REG_A9	Rx Channel 9 Global Config Register A	
1940h	RCHAN_GCFG_REG_A10	Rx Channel 10 Global Config Register A	
1960h	RCHAN_GCFG_REG_A11	Rx Channel 11 Global Config Register A	
1980h	RCHAN_GCFG_REG_A12	Rx Channel 12 Global Config Register A	
19A0h	RCHAN_GCFG_REG_A13	Rx Channel 13 Global Config Register A	
19C0h	RCHAN_GCFG_REG_A14	Rx Channel 14 Global Config Register A	
19E0h	RCHAN_GCFG_REG_A15	Rx Channel 15 Global Config Register A	
PKTDMA TX DMA SCHEDULER REGISTERS			
1C00h	TCHAN_SCHED_CFG_REG0	Tx Channel 0 Scheduler Config Register	See Section 3.14
1C04h	TCHAN_SCHED_CFG_REG1	Tx Channel 1 Scheduler Config Register	
1C08h	TCHAN_SCHED_CFG_REG2	Tx Channel 2 Scheduler Config Register	
1C0Ch	TCHAN_SCHED_CFG_REG3	Tx Channel 3 Scheduler Config Register	
1C10h	TCHAN_SCHED_CFG_REG4	Tx Channel 4 Scheduler Config Register	
1C14h	TCHAN_SCHED_CFG_REG5	Tx Channel 5 Scheduler Config Register	
1C18h	TCHAN_SCHED_CFG_REG6	Tx Channel 6 Scheduler Config Register	
1C1Ch	TCHAN_SCHED_CFG_REG7	Tx Channel 7 Scheduler Config Register	See Section 3.14
1C20h	TCHAN_SCHED_CFG_REG8	Tx Channel 8 Scheduler Config Register	
1C24h	TCHAN_SCHED_CFG_REG9	Tx Channel 9 Scheduler Config Register	
1C28h	TCHAN_SCHED_CFG_REG10	Tx Channel 10 Scheduler Config Register	
1C2Ch	TCHAN_SCHED_CFG_REG11	Tx Channel 11 Scheduler Config Register	
1C30h	TCHAN_SCHED_CFG_REG12	Tx Channel 12 Scheduler Config Register	
1C34h	TCHAN_SCHED_CFG_REG13	Tx Channel 13 Scheduler Config Register	
1C38h	TCHAN_SCHED_CFG_REG14	Tx Channel 14 Scheduler Config Register	
1C3Ch	TCHAN_SCHED_CFG_REG15	Tx Channel 15 Scheduler Config Register	
PKTDMA RX DMA FLOW CONFIGURATION REGISTERS			

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
2000h	RFLOW_CFG_REG_A0	Rx Flow 0 Config Register A	See Section 3.14
2004h	RFLOW_CFG_REG_B0	Rx Flow 0 Config Register B	
2008h	RFLOW_CFG_REG_C0	Rx Flow 0 Config Register C	
200Ch	RFLOW_CFG_REG_D0	Rx Flow 0 Config Register D	
2010h	RFLOW_CFG_REG_E0	Rx Flow 0 Config Register E	
2014h	RFLOW_CFG_REG_F0	Rx Flow 0 Config Register F	
2018h	RFLOW_CFG_REG_G0	Rx Flow 0 Config Register G	
201Ch	RFLOW_CFG_REG_H0	Rx Flow 0 Config Register H	
2020h	RFLOW_CFG_REG_A1	Rx Flow 1 Config Register A	
2024h	RFLOW_CFG_REG_B1	Rx Flow 1 Config Register B	
2028h	RFLOW_CFG_REG_C1	Rx Flow 1 Config Register C	
202Ch	RFLOW_CFG_REG_D1	Rx Flow 1 Config Register D	
2030h	RFLOW_CFG_REG_E1	Rx Flow 1 Config Register E	
2034h	RFLOW_CFG_REG_F1	Rx Flow 1 Config Register F	
2038h	RFLOW_CFG_REG_G1	Rx Flow 1 Config Register G	
203Ch	RFLOW_CFG_REG_H1	Rx Flow 1 Config Register H	
2040h	RFLOW_CFG_REG_A2	Rx Flow 2 Config Register A	
2044h	RFLOW_CFG_REG_B2	Rx Flow 2 Config Register B	
2048h	RFLOW_CFG_REG_C2	Rx Flow 2 Config Register C	
204Ch	RFLOW_CFG_REG_D2	Rx Flow 2 Config Register D	
2050h	RFLOW_CFG_REG_E2	Rx Flow 2 Config Register E	
2054h	RFLOW_CFG_REG_F2	Rx Flow 2 Config Register F	
2058h	RFLOW_CFG_REG_G2	Rx Flow 2 Config Register G	
205Ch	RFLOW_CFG_REG_H2	Rx Flow 2 Config Register H	
2060h	RFLOW_CFG_REG_A3	Rx Flow 3 Config Register A	
2064h	RFLOW_CFG_REG_B3	Rx Flow 3 Config Register B	
2068h	RFLOW_CFG_REG_C3	Rx Flow 3 Config Register C	
206Ch	RFLOW_CFG_REG_D3	Rx Flow 3 Config Register D	
2070h	RFLOW_CFG_REG_E3	Rx Flow 3 Config Register E	
2074h	RFLOW_CFG_REG_F3	Rx Flow 3 Config Register F	
2078h	RFLOW_CFG_REG_G3	Rx Flow 3 Config Register G	
207Ch	RFLOW_CFG_REG_H3	Rx Flow 3 Config Register H	
2080h	RFLOW_CFG_REG_A4	Rx Flow 4 Config Register A	
2084h	RFLOW_CFG_REG_B4	Rx Flow 4 Config Register B	

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
2088h	RFLOW_CFG_REG_C4	Rx Flow 4 Config Register C	See Section 3.14
208Ch	RFLOW_CFG_REG_D4	Rx Flow 4 Config Register D	
2090h	RFLOW_CFG_REG_E4	Rx Flow 4 Config Register E	
2094h	RFLOW_CFG_REG_F4	Rx Flow 4 Config Register F	
2098h	RFLOW_CFG_REG_G4	Rx Flow 4 Config Register G	
209Ch	RFLOW_CFG_REG_H4	Rx Flow 4 Config Register H	
20A0h	RFLOW_CFG_REG_A5	Rx Flow 5 Config Register A	
20A4h	RFLOW_CFG_REG_B5	Rx Flow 5 Config Register B	
20A8h	RFLOW_CFG_REG_C5	Rx Flow 5 Config Register C	
20Ach	RFLOW_CFG_REG_D5	Rx Flow 5 Config Register D	
20B0h	RFLOW_CFG_REG_E5	Rx Flow 5 Config Register E	
20B4h	RFLOW_CFG_REG_F5	Rx Flow 5 Config Register F	
20B8h	RFLOW_CFG_REG_G5	Rx Flow 5 Config Register G	
20BCh	RFLOW_CFG_REG_H5	Rx Flow 5 Config Register H	
20C0h	RFLOW_CFG_REG_A6	Rx Flow 6 Config Register A	
20C4h	RFLOW_CFG_REG_B6	Rx Flow 6 Config Register B	
20C8h	RFLOW_CFG_REG_C6	Rx Flow 6 Config Register C	
20CCh	RFLOW_CFG_REG_D6	Rx Flow 6 Config Register D	
20D0h	RFLOW_CFG_REG_E6	Rx Flow 6 Config Register E	
20D4h	RFLOW_CFG_REG_F6	Rx Flow 6 Config Register F	
20D8h	RFLOW_CFG_REG_G6	Rx Flow 6 Config Register G	
20DCh	RFLOW_CFG_REG_H6	Rx Flow 6 Config Register H	
20E0h	RFLOW_CFG_REG_A7	Rx Flow 7 Config Register A	
20E4h	RFLOW_CFG_REG_B7	Rx Flow 7 Config Register B	
20E8h	RFLOW_CFG_REG_C7	Rx Flow 7 Config Register C	
20ECh	RFLOW_CFG_REG_D7	Rx Flow 7 Config Register D	
20F0h	RFLOW_CFG_REG_E7	Rx Flow 7 Config Register E	
20F4h	RFLOW_CFG_REG_F7	Rx Flow 7 Config Register F	
20F8h	RFLOW_CFG_REG_G7	Rx Flow 7 Config Register G	
20FCh	RFLOW_CFG_REG_H7	Rx Flow 7 Config Register H	
2100h	RFLOW_CFG_REG_A8	Rx Flow 8 Config Register A	
2104h	RFLOW_CFG_REG_B8	Rx Flow 8 Config Register B	
2108h	RFLOW_CFG_REG_C8	Rx Flow 8 Config Register C	
210Ch	RFLOW_CFG_REG_D8	Rx Flow 8 Config Register D	
2110h	RFLOW_CFG_REG_E8	Rx Flow 8 Config Register E	
2114h	RFLOW_CFG_REG_F8	Rx Flow 8 Config Register F	
2118h	RFLOW_CFG_REG_G8	Rx Flow 8 Config Register G	
211Ch	RFLOW_CFG_REG_H8	Rx Flow 8 Config Register H	
2120h	RFLOW_CFG_REG_A9	Rx Flow 9 Config Register A	
2124h	RFLOW_CFG_REG_B9	Rx Flow 9 Config Register B	
2128h	RFLOW_CFG_REG_C9	Rx Flow 9 Config Register C	
212Ch	RFLOW_CFG_REG_D9	Rx Flow 9 Config Register D	
2130h	RFLOW_CFG_REG_E9	Rx Flow 9 Config Register E	
2134h	RFLOW_CFG_REG_F9	Rx Flow 9 Config Register F	

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
2138h	RFLOW_CFG_REG_G9	Rx Flow 9 Config Register G	See Section 3.14
213Ch	RFLOW_CFG_REG_H9	Rx Flow 9 Config Register H	
2140h	RFLOW_CFG_REG_A10	Rx Flow 10 Config Register A	
2144h	RFLOW_CFG_REG_B10	Rx Flow 10 Config Register B	
2148h	RFLOW_CFG_REG_C10	Rx Flow 10 Config Register C	
214Ch	RFLOW_CFG_REG_D10	Rx Flow 10 Config Register D	
2150h	RFLOW_CFG_REG_E10	Rx Flow 10 Config Register E	
2154h	RFLOW_CFG_REG_F10	Rx Flow 10 Config Register F	
2158h	RFLOW_CFG_REG_G10	Rx Flow 10 Config Register G	
215Ch	RFLOW_CFG_REG_H10	Rx Flow 10 Config Register H	
2160h	RFLOW_CFG_REG_A11	Rx Flow 11 Config Register A	
2164h	RFLOW_CFG_REG_B11	Rx Flow 11 Config Register B	
2168h	RFLOW_CFG_REG_C11	Rx Flow 11 Config Register C	
216Ch	RFLOW_CFG_REG_D11	Rx Flow 11 Config Register D	
2170h	RFLOW_CFG_REG_E11	Rx Flow 11 Config Register E	
2174h	RFLOW_CFG_REG_F11	Rx Flow 11 Config Register F	
2178h	RFLOW_CFG_REG_G11	Rx Flow 11 Config Register G	
217Ch	RFLOW_CFG_REG_H11	Rx Flow 11 Config Register H	
2180h	RFLOW_CFG_REG_A12	Rx Flow 12 Config Register A	
2184h	RFLOW_CFG_REG_B12	Rx Flow 12 Config Register B	
2188h	RFLOW_CFG_REG_C12	Rx Flow 12 Config Register C	
218Ch	RFLOW_CFG_REG_D12	Rx Flow 12 Config Register D	
2190h	RFLOW_CFG_REG_E12	Rx Flow 12 Config Register E	
2194h	RFLOW_CFG_REG_F12	Rx Flow 12 Config Register F	
2198h	RFLOW_CFG_REG_G12	Rx Flow 12 Config Register G	
219Ch	RFLOW_CFG_REG_H12	Rx Flow 12 Config Register H	
21A0h	RFLOW_CFG_REG_A13	Rx Flow 13 Config Register A	
21A4h	RFLOW_CFG_REG_B13	Rx Flow 13 Config Register B	
21A8h	RFLOW_CFG_REG_C13	Rx Flow 13 Config Register C	
21Ach	RFLOW_CFG_REG_D13	Rx Flow 13 Config Register D	
21B0h	RFLOW_CFG_REG_E13	Rx Flow 13 Config Register E	
21B4h	RFLOW_CFG_REG_F13	Rx Flow 13 Config Register F	
21B8h	RFLOW_CFG_REG_G13	Rx Flow 13 Config Register G	
21BCh	RFLOW_CFG_REG_H13	Rx Flow 13 Config Register H	
21C0h	RFLOW_CFG_REG_A14	Rx Flow 14 Config Register A	
21C4h	RFLOW_CFG_REG_B14	Rx Flow 14 Config Register B	
21C8h	RFLOW_CFG_REG_C14	Rx Flow 14 Config Register C	
21CCh	RFLOW_CFG_REG_D14	Rx Flow 14 Config Register D	
21D0h	RFLOW_CFG_REG_E14	Rx Flow 14 Config Register E	
21D4h	RFLOW_CFG_REG_F14	Rx Flow 14 Config Register F	
21D8h	RFLOW_CFG_REG_G14	Rx Flow 14 Config Register G	
21DCh	RFLOW_CFG_REG_H14	Rx Flow 14 Config Register H	
21E0h	RFLOW_CFG_REG_A15	Rx Flow 15 Config Register A	
21E4h	RFLOW_CFG_REG_B15	Rx Flow 15 Config Register B	

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
21E8h	RFLOW_CFG_REG_C15	Rx Flow 15 Config Register C	See Section 3.14
21ECh	RFLOW_CFG_REG_D15	Rx Flow 15 Config Register D	
21F0h	RFLOW_CFG_REG_E15	Rx Flow 15 Config Register E	
21F4h	RFLOW_CFG_REG_F15	Rx Flow 15 Config Register F	
21F8h	RFLOW_CFG_REG_G15	Rx Flow 15 Config Register G	
21FCh	RFLOW_CFG_REG_H15	Rx Flow 15 Config Register H	
2200h	RFLOW_CFG_REG_A16	Rx Flow 16 Config Register A	
2204h	RFLOW_CFG_REG_B16	Rx Flow 16 Config Register B	
2208h	RFLOW_CFG_REG_C16	Rx Flow 16 Config Register C	
220Ch	RFLOW_CFG_REG_D16	Rx Flow 16 Config Register D	
2210h	RFLOW_CFG_REG_E16	Rx Flow 16 Config Register E	
2214h	RFLOW_CFG_REG_F16	Rx Flow 16 Config Register F	
2218h	RFLOW_CFG_REG_G16	Rx Flow 16 Config Register G	
221Ch	RFLOW_CFG_REG_H16	Rx Flow 16 Config Register H	
2220h	RFLOW_CFG_REG_A17	Rx Flow 17 Config Register A	
2224h	RFLOW_CFG_REG_B17	Rx Flow 17 Config Register B	
2228h	RFLOW_CFG_REG_C17	Rx Flow 17 Config Register C	
222Ch	RFLOW_CFG_REG_D17	Rx Flow 17 Config Register D	
2230h	RFLOW_CFG_REG_E17	Rx Flow 17 Config Register E	
2234h	RFLOW_CFG_REG_F17	Rx Flow 17 Config Register F	
2238h	RFLOW_CFG_REG_G17	Rx Flow 17 Config Register G	
223Ch	RFLOW_CFG_REG_H17	Rx Flow 17 Config Register H	
2240h	RFLOW_CFG_REG_A18	Rx Flow 18 Config Register A	
2244h	RFLOW_CFG_REG_B18	Rx Flow 18 Config Register B	
2248h	RFLOW_CFG_REG_C18	Rx Flow 18 Config Register C	
224Ch	RFLOW_CFG_REG_D18	Rx Flow 18 Config Register D	
2250h	RFLOW_CFG_REG_E18	Rx Flow 18 Config Register E	
2254h	RFLOW_CFG_REG_F18	Rx Flow 18 Config Register F	
2258h	RFLOW_CFG_REG_G18	Rx Flow 18 Config Register G	
225Ch	RFLOW_CFG_REG_H18	Rx Flow 18 Config Register H	
2260h	RFLOW_CFG_REG_A19	Rx Flow 19 Config Register A	
2264h	RFLOW_CFG_REG_B19	Rx Flow 19 Config Register B	
2268h	RFLOW_CFG_REG_C19	Rx Flow 19 Config Register C	
226Ch	RFLOW_CFG_REG_D19	Rx Flow 19 Config Register D	
2270h	RFLOW_CFG_REG_E19	Rx Flow 19 Config Register E	
2274h	RFLOW_CFG_REG_F19	Rx Flow 19 Config Register F	
2278h	RFLOW_CFG_REG_G19	Rx Flow 19 Config Register G	
227Ch	RFLOW_CFG_REG_H19	Rx Flow 19 Config Register H	
CSR/CAR REGISTERS			
B000h	DEV_ID	Device Identity CAR	Section 3.15.1
B004h	DEV_INFO	Device Information CAR	Section 3.15.2
B008h	ASBLY_ID	Assembly Identity CAR	Section 3.15.3
B00Ch	ASBLY_INFO	Assembly Information CAR	Section 3.15.4

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
B010h	PE_FEAT	Processing Element Features CAR	Section 3.15.5
B014h	SW_PORT	Switch Port Information CAR	Section 3.15.6
B018h	SRC_OP	Source Operations CAR	Section 3.15.7
B01Ch	DEST_OP	Destination Operations CAR	Section 3.15.8
B03Ch	DS_INFO	Data Streaming Information CSR	Section 3.15.9
B048h	DS_LL_CTL	Data Streaming Logical Layer Control CSR	Section 3.15.10
B04Ch	PE_LL_CTL	Processing Element Logical Layer Control CSR	Section 3.15.11
B058h	LCL_CFG_HBAR	Local Configuration Space Base Address 0 CSR	Section 3.15.13
B05Ch	LCL_CFG_BAR	Local Configuration Space Base Address 1 CSR	Section 3.15.13
B060h	BASE_ID	Base Device ID CSR	Section 3.15.14
B068h	HOST_BASE_ID_LOCK	Host Base Device ID Lock CSR	Section 3.15.15
B06Ch	COMP_TAG	Component Tag CSR	Section 3.15.16
B100h	SP_MB_HEAD	1x/4x LP-Serial Port Maintenance Block Header	Section 3.15.17
B120h	SP_LT_CTL	Port Link Time-Out Control CSR	Section 3.15.18
B124h	SP_RT_CTL	Port Response Time-Out Control CSR	Section 3.15.19
B13Ch	SP_GEN_CTL	Port General Control CSR	Section 3.15.20
B140h	SP0_LM_REQ	Port 0 Link Maintenance Request CSR	Section 3.15.21
B144h	SP0_LM_RESP	Port 0 Link Maintenance Response CSR	Section 3.15.22
B148h	SP0_ACKID_STAT	Port 0 Local AckID Status CSR	Section 3.15.23
B154h	SP0_CTL2	Port 0 Control 2 CSR	Section 3.15.24
B158h	SP0_ERR_STAT	Port 0 Error and Status CSR	Section 3.15.25
B15Ch	SP0_CTL	Port 0 Control CSR	Section 3.15.26
B160h	SP1_LM_REQ	Port 1 Link Maintenance Request CSR	Section 3.15.21
B164h	SP1_LM_RESP	Port 1 Link Maintenance Response CSR	Section 3.15.22
B168h	SP1_ACKID_STAT	Port 1 Local AckID Status CSR	Section 3.15.23
B174h	SP1_CTL2	Port 1 Control 2 CSR	Section 3.15.24
B178h	SP1_ERR_STAT	Port 1 Error and Status CSR	Section 3.15.25

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
B17Ch	SP1_CTL	Port 1 Control CSR	Section 3.15.26
B180h	SP2_LM_REQ	Port 2 Link Maintenance Request CSR	Section 3.15.21
B184h	SP2_LM_RESP	Port 2 Link Maintenance Response CSR	Section 3.15.22
B188h	SP2_ACKID_STAT	Port 2 Local AckID Status CSR	Section 3.15.23
B194h	SP2_CTL2	Port 2 Control 2 CSR	Section 3.15.24
B198h	SP2_ERR_STAT	Port 2 Error and Status CSR	Section 3.15.25
B19Ch	SP2_CTL	Port 2 Control CSR	Section 3.15.26
B1A0h	SP3_LM_REQ	Port 3 Link Maintenance Request CSR	Section 3.15.21
B1A4h	SP3_LM_RESP	Port 3 Link Maintenance Response CSR	Section 3.15.22
B1A8h	SP3_ACKID_STAT	Port 3 Local AckID Status CSR	Section 3.15.23
B1B4h	SP3_CTL2	Port 3 Control 2 CSR	Section 3.15.24
B1B8h	SP3_ERR_STAT	Port 3 Error and Status CSR	Section 3.15.25
B1BCh	SP3_CTL	Port 3 Control CSR	Section 3.15.26
ERROR MANAGEMENT REGISTERS			
C000h	ERR_RPT_BH	Error Reporting Block Header	Section 3.16.1
C008h	ERR_DET	Logical/Transport Layer Error Detect CSR	Section 3.16.2
C00Ch	ERR_EN	Logical/Transport Layer Error Enable CSR	Section 3.16.3
C010h	H_ADDR_CAPT	Logical/Transport Layer High Address Capture CSR	Section 3.16.4
C014h	ADDR_CAPT	Logical/Transport Layer Address Capture CSR	Section 3.16.5
C018h	ID_CAPT	Logical/Transport Layer Device ID Capture CSR	Section 3.16.6
C01Ch	CTRL_CAPT	Logical/Transport Layer Control Capture CSR	Section 3.16.7
C028h	PW_TGT_ID	Port-Write Target Device ID CSR	Section 3.16.8
C040h	SP0_ERR_DET	Port 0 Error Detect CSR	Section 3.16.9
C044h	SP0_RATE_EN	Port 0 Error Enable CSR	Section 3.16.10
C048h	SP0_ERR_ATTR_CAPT_DBG0	Port 0 Attributes Error Capture CSR	Section 3.16.11
C04Ch	SP0_ERR_CAPT_0_DBG1	Port 0 Packet/Control Symbol Error Capture CSR0	Section 3.16.12
C050h	SP0_ERR_CAPT_1_DBG2	Port 0 Packet Error Capture CSR1	Section 3.16.13
C054h	SP0_ERR_CAPT_2_DBG3	Port 0 Packet Error Capture CSR2	Section 3.16.14

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
C058h	SP0_ERR_CAPT_3_DBG4	Port 0 Packet Error Capture CSR3	Section 3.16.15
C068h	SP0_ERR_RATE	Port 0 Error Rate CSR	Section 3.16.16
C06Ch	SP0_ERR_THRESH	Port 0 Error Rate Threshold CSR	Section 3.16.17
C080h	SP1_ERR_DET	Port 1 Error Detect CSR	Section 3.16.9
C084h	SP1_RATE_EN	Port 1 Error Enable CSR	Section 3.16.10
C088h	SP1_ERR_ATTR_CAPT	Port 1 Attributes Error Capture CSR	Section 3.16.11
C08Ch	SP1_ERR_CAPT_0	Port 1 Packet/Control Symbol Error Capture CSR0	Section 3.16.12
C090h	SP1_ERR_CAPT_1	Port 1 Packet Error Capture CSR1	Section 3.16.13
C094h	SP1_ERR_CAPT_2	Port 1 Packet Error Capture CSR2	Section 3.16.14
C098h	SP1_ERR_CAPT_3	Port 1 Packet Error Capture CSR3	Section 3.16.15
C0A8h	SP1_ERR_RATE	Port 1 Error Rate CSR	Section 3.16.16
C0ACh	SP1_ERR_THRESH	Port 1 Error Rate Threshold CSR	Section 3.16.17
C0C0h	SP2_ERR_DET	Port 2 Error Detect CSR	Section 3.16.9
C0C4h	SP2_RATE_EN	Port 2 Error Enable CSR	Section 3.16.10
C0C8h	SP2_ERR_ATTR_CAPT	Port 2 Attributes Error Capture CSR	Section 3.16.11
C0CCh	SP2_ERR_CAPT_0	Port 2 Packet/Control Symbol Error Capture CSR0	Section 3.16.12
C0D0h	SP2_ERR_CAPT_1	Port 2 Packet Error Capture CSR1	Section 3.16.13
C0D4h	SP2_ERR_CAPT_2	Port 2 Packet Error Capture CSR2	Section 3.16.14
C0D8h	SP2_ERR_CAPT_3	Port 2 Packet Error Capture CSR3	Section 3.16.15
C0E8h	SP2_ERR_RATE	Port 2 Error Rate CSR	Section 3.16.16
C0ECh	SP2_ERR_THRESH	Port 2 Error Rate Threshold CSR	Section 3.16.17
C100h	SP3_ERR_DET	Port 3 Error Detect CSR	Section 3.16.9
C104h	SP3_RATE_EN	Port 3 Error Enable CSR	Section 3.16.10
C108h	SP3_ERR_ATTR_CAPT	Port 3 Attributes Error Capture CSR	Section 3.16.11
C10Ch	SP3_ERR_CAPT_0	Packet/Control Symbol Error Capture CSR0	Section 3.16.12
C110h	SP3_ERR_CAPT_1	Port 3 Packet Error Capture CSR1	Section 3.16.13
C114h	SP3_ERR_CAPT_2	Port 3 Packet Error Capture CSR2	Section 3.16.14

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
C118h	SP3_ERR_CAPT_3	Port 3 Packet Error Capture CSR3	Section 3.16.15
C128h	SP3_ERR_RATE	Port 3 Error Rate CSR	Section 3.16.16
C12Ch	SP3_ERR_THRESH	Port 3 Error Rate Threshold CSR	Section 3.16.17
E010h	LANE0_STAT0	Lane 0 Status 0 CSR	Section 3.17.1
E014h	LANE0_STAT1	Lane 0 Status 1 CSR	Section 3.17.2
E030h	LANE1_STAT0	Lane 1 Status 0 CSR	Section 3.17.1
E034h	LANE1_STAT1	Lane 1 Status 1 CSR	Section 3.17.2
E050h	LANE2_STAT0	Lane 2 Status 0 CSR	Section 3.17.1
E054h	LANE2_STAT1	Lane 2 Status 1 CSR	Section 3.17.2
E070h	LANE3_STAT0	Lane 3 Status 0 CSR	Section 3.17.1
E074h	LANE3_STAT1	Lane 3 Status 1 CSR	Section 3.17.2
1B000h	PLM_BH	PLM Block Header Register	Section 3.18.1
1B080h	PLM_SP0_IMP_SPEC_CTL	Port 0-3 implementation specific control register	Section 3.18.2
1B084h	RIO_PLM_SP0_PWDN_CTL	Port 0-3 Powerdown control register	Section 3.18.3
1B090h	PLM_SP0_STATUS	Port 0-3 event status register	Section 3.18.4
1B094h	PLM_SP0_INT_ENABLE	PLM port 0-3 interrupt enable register	Section 3.18.5
1B098h	PLM_SP0_PW_ENABLE	PLM port 0-3 port-write enable register	Section 3.18.6
1B09Ch	PLM_SP0_EVENT_GEN	PLM port 0-3 event generate register	Section 3.18.7
1B0A0h	PLM_SP0_ALL_INT_EN	Port 0-3 all interrupts enable register	Section 3.18.8
1B0A4h	PLM_SP0_ALL_PW_EN	Port 0-3 all port-writes enable register	Section 3.18.9
1B0B0h	PLM_SP0_PATH_CTL	Port 0-3 Path Control register	Section 3.18.10
1B0B4h	PLM_SP0_DISCOVERY_TIMER	Port 0-3 discovery timer register	Section 3.18.11
1B0B8h	PLM_SP0_SILENCE_TIMER	Port 0-3 silence timer register	Section 3.18.12
1B0BCh	PLM_SP0_VMIN_EXP	Port 0-3 Vmin exponent register	Section 3.18.13
1B0C0h	PLM_SP0_POL_CTL	Port 0-3 lane polarity control register	Section 3.18.14
1B0C8h	PLM_SP0_DENIAL_CTL	Port 0-3 packet denial control register	Section 3.18.15
1B0D0h	PLM_SP0_RCVD_MECS	Port 0-3 received MECS status register	Section 3.18.16

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B0D8h	PLM_SP0_MECS_FWD	Port 0-3 MECS forwarding register	Section 3.18.17
1B0E0h	PLM_SP0_LONG_CS_TX1	Port 0-3 long control symbol transmit 1	Section 3.18.18
1B0E4h	PLM_SP0_LONG_CS_TX2	Port 0-3 long control symbol transmit 2	Section 3.18.19
1B100h	PLM_SP1_IMP_SPEC_CTL	Port 0-3 implementation specific control register	Section 3.18.2
1B104h	RIO_PLM_SP1_PWDN_CTL	Port 0-3 Powerdown control register	Section 3.18.3
1B110h	PLM_SP1_STATUS	Port 0-3 event status register	Section 3.18.4
1B114h	PLM_SP1_INT_ENABLE	PLM port 0-3 interrupt enable register	Section 3.18.5
1B118h	PLM_SP1_PW_ENABLE	PLM port 0-3 port-write enable register	Section 3.18.6
1B11Ch	PLM_SP1_EVENT_GEN	PLM port 0-3 event generate register	Section 3.18.7
1B120h	PLM_SP1_ALL_INT_EN	Port 0-3 all interrupts enable register	Section 3.18.8
1B124h	PLM_SP1_ALL_PW_EN	Port 0-3 all port-writes enable register	Section 3.18.9
1B130h	PLM_SP1_PATH_CTL	Port 0-3 Path Control register	Section 3.18.10
1B134h	PLM_SP1_DISCOVERY_TIMER	Port 0-3 discovery timer register	Section 3.18.11
1B138h	PLM_SP1_SILENCE_TIMER	Port 0-3 silence timer register	Section 3.18.12
1B13Ch	PLM_SP1_VMIN_EXP	Port 0-3 Vmin exponent register	Section 3.18.13
1B140h	PLM_SP1_POL_CTL	Port 0-3 lane polarity control register	Section 3.18.14
1B148h	PLM_SP1_DENIAL_CTL	Port 0-3 packet denial control register	Section 3.18.15
1B150h	PLM_SP1_RCVD_MECS	Port 0-3 received MECS status register	Section 3.18.16
1B158h	PLM_SP1_MECS_FWD	Port 0-3 MECS forwarding register	Section 3.18.17
1B160h	PLM_SP1_LONG_CS_TX1	Port 0-3 long control symbol transmit 1	Section 3.18.18
1B164h	PLM_SP1_LONG_CS_TX2	Port 0-3 long control symbol transmit 2	Section 3.18.19
1B180h	PLM_SP2_IMP_SPEC_CTL	Port 0-3 implementation specific control register	Section 3.18.2
1B184h	PLM_SP2_PWDN_CTL	Port 0-3 Powerdown control register	Section 3.18.3
1B190h	PLM_SP2_STATUS	Port 0-3 event status register	Section 3.18.4
1B194h	PLM_SP2_INT_ENABLE	PLM port 0-3 interrupt enable register	Section 3.18.5
1B198h	PLM_SP2_PW_ENABLE	PLM port 0-3 port-write enable register	Section 3.18.6
1B19Ch	PLM_SP2_EVENT_GEN	PLM port 0-3 event generate register	Section 3.18.7

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B1A0h	PLM_SP2_ALL_INT_EN	Port 0-3 all interrupts enable register	Section 3.18.8
1B1A4h	PLM_SP2_ALL_PW_EN	Port 0-3 all port-writes enable register	Section 3.18.9
1B1B0h	PLM_SP2_PATH_CTL	Port 0-3 Path Control register	Section 3.18.10
1B1B4h	PLM_SP2_DISCOVERY_TIMER	Port 0-3 discovery timer register	Section 3.18.11
1B1B8h	PLM_SP2_SILENCE_TIMER	Port 0-3 silence timer register	Section 3.18.12
1B1BCh	PLM_SP2_VMIN_EXP	Port 0-3 Vmin exponent register	Section 3.18.13
1B1C0h	PLM_SP2_POL_CTL	Port 0-3 lane polarity control register	Section 3.18.14
1B1C8h	PLM_SP2_DENIAL_CTL	Port 0-3 packet denial control register	Section 3.18.15
1B1D0h	PLM_SP2_RCVD_MECS	Port 0-3 received MECS status register	Section 3.18.16
1B1D8h	PLM_SP2_MECS_FWD	Port 0-3 MECS forwarding register	Section 3.18.17
1B1E0h	PLM_SP2_LONG_CS_TX1	Port 0-3 long control symbol transmit 1	Section 3.18.18
1B1E4h	PLM_SP2_LONG_CS_TX2	Port 0-3 long control symbol transmit 2	Section 3.18.19
1B200h	PLM_SP3_IMP_SPEC_CTL	Port 0-3 implementation specific control register	Section 3.18.2
1B204h	PLM_SP3_PWDN_CTL	Port 0-3 Powerdown control register	Section 3.18.3
1B210h	PLM_SP3_STATUS	Port 0-3 event status register	Section 3.18.4
1B214h	PLM_SP3_INT_ENABLE	PLM port 0-3 interrupt enable register	Section 3.18.5
1B218h	PLM_SP3_PW_ENABLE	PLM port 0-3 port-write enable register	Section 3.18.6
1B21Ch	PLM_SP3_EVENT_GEN	PLM port 0-3 event generate register	Section 3.18.7
1B220h	PLM_SP3_ALL_INT_EN	Port 0-3 all interrupts enable register	Section 3.18.8
1B224h	PLM_SP3_ALL_PW_EN	Port 0-3 all port-writes enable register	Section 3.18.9
1B230h	PLM_SP3_PATH_CTL	Port 0-3 Path Control register	Section 3.18.10
1B234h	PLM_SP3_DISCOVERY_TIMER	Port 0-3 discovery timer register	Section 3.18.11
1B238h	PLM_SP3_SILENCE_TIMER	Port 0-3 silence timer register	Section 3.18.12
1B23Ch	PLM_SP3_VMIN_EXP	Port 0-3 Vmin exponent register	Section 3.18.13
1B240h	PLM_SP3_POL_CTL	Port 0-3 lane polarity control register	Section 3.18.14
1B248h	PLM_SP3_DENIAL_CTL	Port 0-3 packet denial control register	Section 3.18.15
1B250h	PLM_SP3_RCVD_MECS	Port 0-3 received MECS status register	Section 3.18.16

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B258h	PLM_SP3_MECS_FWD	Port 0-3 MECS forwarding register	Section 3.18.17
1B260h	PLM_SP3_LONG_CS_TX1	Port 0-3 long control symbol transmit 1	Section 3.18.18
1B264h	PLM_SP3_LONG_CS_TX2	Port 0-3 long control symbol transmit 2	Section 3.18.19
1B300h	TLM_BH	TLM block header register	Section 3.18.20
1B380h	TLM_SP0_CONTROL	Port 0-3 control register	Section 3.18.21
1B390h	TLM_SP0_STATUS	TLM port 0-3 event status register	Section 3.18.22
1B394h	TLM_SP0_INT_ENABLE	TLM port 0-3 interrupt enable register	Section 3.18.23
1B398h	TLM_SP0_PW_ENABLE	TLM port 0-3 port-write enable register	Section 3.18.24
1B39Ch	TLM_SP0_EVENT_GEN	TLM port 0-3 event generate register	Section 3.18.25
1B3A0h	TLM_SP0_BRR_0_CTL	TLM port 0-3 base routing register 0 Control (Not supported by TI)	Section 3.18.26
1B3A4h	TLM_SP0_BRR_0_PATTERN_MATCH	TLM port 0-3 BRR 0 pattern & match (Not supported by TI)	Section 3.18.27
1B3B0h	TLM_SP0_BRR_1_CTL	TLM port 0-3 base routing register 1 Control	Section 3.18.28
1B3B4h	TLM_SP0_BRR_1_PATTERN_MATCH	TLM port 0-3 BRR 1 pattern & match	Section 3.18.29
1B3C0h	TLM_SP0_BRR_2_CTL	TLM port 0-3 base routing register 2 Control	Section 3.18.30
1B3C4h	TLM_SP0_BRR_2_PATTERN_MATCH	TLM port 0-3 BRR 2 pattern & match	Section 3.18.31
1B3D0h	TLM_SP0_BRR_3_CTL	TLM port 0-3 base routing register 3 Control	Section 3.18.32
1B3D4h	TLM_SP0_BRR_3_PATTERN_MATCH	TLM port 0-3 BRR 3 pattern & match	Section 3.18.33
1B400h	TLM_SP1_CONTROL	Port 0-3 control register	Section 3.18.21
1B410h	TLM_SP1_STATUS	TLM port 0-3 event status register	Section 3.18.22
1B414h	TLM_SP1_INT_ENABLE	TLM port 0-3 interrupt enable register	Section 3.18.23
1B418h	TLM_SP1_PW_ENABLE	TLM port 0-3 port-write enable register	Section 3.18.24
1B41Ch	TLM_SP1_EVENT_GEN	TLM port 0-3 event generate register	Section 3.18.25
1B420h	TLM_SP1_BRR_0_CTL	TLM port 0-3 base routing register 0 Control	Section 3.18.26
1B424h	TLM_SP1_BRR_0_PATTERN_MATCH	TLM port 0-3 BRR 0 pattern & match	Section 3.18.27
1B430h	TLM_SP1_BRR_1_CTL	TLM port 0-3 base routing register 1 Control	Section 3.18.28
1B434h	TLM_SP1_BRR_1_PATTERN_MATCH	TLM port 0-3 BRR 1 pattern & match	Section 3.18.29
1B440h	TLM_SP1_BRR_2_CTL	TLM port 0-3 base routing register 2 Control	Section 3.18.30

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B444h	TLM_SP1_BRR_2_PATTERN_MATCH	TLM port 0-3 BRR 2 pattern & match	Section 3.18.31
1B450h	TLM_SP1_BRR_3_CTL	TLM port 0-3 base 3 Control	Section 3.18.32
1B454h	TLM_SP1_BRR_3_PATTERN_MATCH	TLM port 0-3 BRR 3 pattern & match	Section 3.18.33
1B480h	TLM_SP2_CONTROL	Port 0-3 control register	Section 3.18.21
1B490h	TLM_SP2_STATUS	TLM port 0-3 event status register	Section 3.18.22
1B494h	TLM_SP2_INT_ENABLE	TLM port 0-3 interrupt enable register	Section 3.18.23
1B498h	TLM_SP2_PW_ENABLE	TLM port 0-3 port-write enable register	Section 3.18.24
1B49Ch	TLM_SP2_EVENT_GEN	TLM port 0-3 event generate register	Section 3.18.25
1B4A0h	TLM_SP2_BRR_0_CTL	TLM port 0-3 base routing register 0 Control	Section 3.18.26
1B4A4h	TLM_SP2_BRR_0_PATTERN_MATCH	TLM port 0-3 BRR 0 pattern & match	Section 3.18.27
1B4B0h	TLM_SP2_BRR_1_CTL	TLM port 0-3 base routing register 1 Control	Section 3.18.28
1B4B4h	TLM_SP2_BRR_1_PATTERN_MATCH	TLM port 0-3 BRR 1 pattern & match	Section 3.18.29
1B4C0h	TLM_SP2_BRR_2_CTL	TLM port 0-3 base routing register 2 Control	Section 3.18.30
1B4C4h	TLM_SP2_BRR_2_PATTERN_MATCH	TLM port 0-3 BRR 2 pattern & match	Section 3.18.31
1B4D0h	TLM_SP2_BRR_3_CTL	TLM port 0-3 base routing register 3 Control	Section 3.18.32
1B4D4h	TLM_SP2_BRR_3_PATTERN_MATCH	TLM port 0-3 BRR 3 pattern & match	Section 3.18.33
1B500h	TLM_SP3_CONTROL	Port 0-3 control register	Section 3.18.21
1B510h	TLM_SP3_STATUS	TLM port 0-3 event status register	Section 3.18.22
1B514h	TLM_SP3_INT_ENABLE	TLM port 0-3 interrupt enable register	Section 3.18.23
1B518h	TLM_SP3_PW_ENABLE	TLM port 0-3 port-write enable register	Section 3.18.24
1B51Ch	TLM_SP3_EVENT_GEN	TLM port 0-3 event generate register	Section 3.18.25
1B520h	TLM_SP3_BRR_0_CTL	TLM port 0-3 base routing register 0 Control	Section 3.18.26
1B524h	TLM_SP3_BRR_0_PATTERN_MATCH	TLM port 0-3 BRR 0 pattern & match	Section 3.18.27
1B530h	TLM_SP3_BRR_1_CTL	TLM port 0-3 base routing register 1 Control	Section 3.18.28
1B534h	TLM_SP3_BRR_1_PATTERN_MATCH	TLM port 0-3 BRR 1 pattern & match	Section 3.18.29
1B540h	TLM_SP3_BRR_2_CTL	TLM port 0-3 base routing register 2 Control	Section 3.18.30
1B544h	TLM_SP3_BRR_2_PATTERN_MATCH	TLM port 0-3 BRR 2 pattern & match	Section 3.18.31

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B550h	TLM_SP3_BRR_3_CTL	TLM port 0-3 base routing register 3 Control	Section 3.18.32
1B554h	TLM_SP3_BRR_3_PATTERN_MATCH	TLM port 0-3 BRR 3 pattern & match	Section 3.18.33
1B600h	PBM_BH	Packet buffer module header register	Section 3.18.34
1B680h	PBM_SP0_CONTROL	PBM port 0-3 control register	Section 3.18.35
1B690h	PBM_SP0_STATUS	PBM port 0-3 status register	Section 3.18.36
1B694h	PBM_SP0_INT_ENABLE	PBM port 0-3 interrupt enable register	Section 3.18.37
1B698h	PBM_SP0_PW_ENABLE	PBM port 0-3 port-write enable register	Section 3.18.38
1B69Ch	PBM_SP0_EVENT_GEN	PBM port 0-3 event generate register	Section 3.18.39
1B6A0h	PBM_SP0_IG_RESOURCES	PBM port 0-3 ingress resources register	Section 3.18.40
1B6A4h	PBM_SP0_EG_RESOURCES	PBM port 0-3 egress resources register	Section 3.18.41
1B6B0h	PBM_SP0_IG_WATERMARK0	PBM port 0-3 ingress watermarks 0 register	Section 3.18.42
1B6B4h	PBM_SP0_IG_WATERMARK1	PBM port 0-3 ingress watermarks 1 register	Section 3.18.43
1B6B8h	PBM_SP0_IG_WATERMARK2	PBM port 0-3 ingress watermarks 2 register	Section 3.18.44
1B6BCh	PBM_SP0_IG_WATERMARK3	PBM port 0-3 ingress watermarks 3 register	Section 3.18.45
1B700h	PBM_SP1_CONTROL	PBM port 0-3 control register	Section 3.18.35
1B710h	PBM_SP1_STATUS	PBM port 0-3 status register	Section 3.18.36
1B714h	PBM_SP1_INT_ENABLE	PBM port 0-3 interrupt enable register	Section 3.18.37
1B718h	PBM_SP1_PW_ENABLE	PBM port 0-3 port-write enable register	Section 3.18.38
1B71Ch	PBM_SP1_EVENT_GEN	PBM port 0-3 event generate register	Section 3.18.39
1B720h	PBM_SP1_IG_RESOURCES	PBM port 0-3 ingress resources register	Section 3.18.40
1B724h	PBM_SP1_EG_RESOURCES	PBM port 0-3 egress resources register	Section 3.18.41
1B730h	PBM_SP1_IG_WATERMARK0	PBM port 0-3 ingress watermarks 0 register	Section 3.18.42
1B734h	PBM_SP1_IG_WATERMARK1	PBM port 0-3 ingress watermarks 1 register	Section 3.18.43
1B738h	PBM_SP1_IG_WATERMARK2	PBM port 0-3 ingress watermarks 2 register	Section 3.18.44
1B73Ch	PBM_SP1_IG_WATERMARK3	PBM port 0-3 ingress watermarks 3 register	Section 3.18.45
1B780h	PBM_SP2_CONTROL	PBM port 0-3 control register	Section 3.18.35
1B790h	PBM_SP2_STATUS	PBM port 0-3 status register	Section 3.18.36

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B794h	PBM_SP2_INT_ENABLE	PBM port 0-3 interrupt enable register	Section 3.18.37
1B798h	PBM_SP2_PW_ENABLE	PBM port 0-3 port-write enable register	Section 3.18.38
1B79Ch	PBM_SP2_EVENT_GEN	PBM port 0-3 event generate register	Section 3.18.39
1B7A0h	PBM_SP2_IG_RESOURCES	PBM port 0-3 ingress resources register	Section 3.18.40
1B7A4h	PBM_SP2_EG_RESOURCES	PBM port 0-3 egress resources register	Section 3.18.41
1B7B0h	PBM_SP2_IG_WATERMARK0	PBM port 0-3 ingress watermarks 0 register	Section 3.18.42
1B7B4h	PBM_SP2_IG_WATERMARK1	PBM port 0-3 ingress watermarks 1 register	Section 3.18.43
1B7B8h	PBM_SP2_IG_WATERMARK2	PBM port 0-3 ingress watermarks 2 register	Section 3.18.44
1B7BCh	PBM_SP2_IG_WATERMARK3	PBM port 0-3 ingress watermarks 3 register	Section 3.18.45
1B800h	PBM_SP3_CONTROL	PBM port 0-3 control register	Section 3.18.35
1B810h	PBM_SP3_STATUS	PBM port 0-3 status register	Section 3.18.36
1B814h	PBM_SP3_INT_ENABLE	PBM port 0-3 interrupt enable register	Section 3.18.37
1B818h	PBM_SP3_PW_ENABLE	PBM port 0-3 port-write enable register	Section 3.18.38
1B81Ch	PBM_SP3_EVENT_GEN	PBM port 0-3 event generate register	Section 3.18.39
1B820h	PBM_SP3_IG_RESOURCES	PBM port 0-3 ingress resources register	Section 3.18.40
1B824h	PBM_SP3_EG_RESOURCES	PBM port 0-3 egress resources register	Section 3.18.41
1B830h	PBM_SP3_IG_WATERMARK0	PBM port 0-3 ingress watermarks 0 register	Section 3.18.42
1B834h	PBM_SP3_IG_WATERMARK1	PBM port 0-3 ingress watermarks 1 register	Section 3.18.43
1B838h	PBM_SP3_IG_WATERMARK2	PBM port 0-3 ingress watermarks 2 register	Section 3.18.44
1B83Ch	PBM_SP3_IG_WATERMARK3	PBM port 0-3 ingress watermarks 3 register	Section 3.18.45
1B900h	EM_BH	IDT-specific event management block header	Section 3.18.46
1B910h	EM_INT_STAT	Event management interrupt status register	Section 3.18.47
1B914h	EM_INT_ENABLE	Event management interrupts enable register	Section 3.18.48
1B918h	EM_INT_PORT_STAT	Event management interrupt port status register	Section 3.18.49
1B920h	EM_PW_STAT	Event management port-write status register	Section 3.18.50
1B924h	EM_PW_EN	Event management port-write enable register	Section 3.18.51
1B928h	EM_PW_PORT_STAT	Event management port-write port status register	Section 3.18.52

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1B930h	EM_DEV_INT_EN	Event management device interrupt enable register	Section 3.18.53
1B934h	EM_DEV_PW_EN	Event management device port-write enable register	Section 3.18.54
1B93Ch	EM_MECS_STAT	Event management MECS status register	Section 3.18.55
1B940h	EM_MECS_INT_EN	Event management MECS interrupt enable register	Section 3.18.56
1B944h	EM_MECS_CAP_EN	Event management MECSCapture Out register	Section 3.18.57
1B948h	EM_MECS_TRIG_EN	Event management MECS Trigger In register	Section 3.18.58
1B94Ch	EM_MECS_REQ	Event management MECS request register	Section 3.18.59
1B950h	EM_MECS_PORT_STAT	Event management MECS port status register	Section 3.18.60
	EM_MECS_EVENT_GEN	Event management MECS event generate register	Section 3.18.61
1B960h	EM_RST_PORT_STAT	Event management reset request port status register	Section 3.18.62
1B968h	EM_RST_INT_EN	Event management reset request interrupt enable register	Section 3.18.63
1B970h	EM_RST_PW_EN	Event management reset request port-write enable register	Section 3.18.64
1BA00h	PW_BH	IDT-specific RIO port-write block header	Section 3.18.65
1BA04h	PW_CTL	RIO port-write control register	Section 3.18.66
1BA08h	PW_ROUTE	RIO port-write routing register	Section 3.18.67
1BA10h	PW_RX_STAT	RIO port-write reception status CSR	Section 3.18.68
1BA14h	RIO_PW_RX_EVENT_GEN	RIO Port-Write Reception Event Generate Register	Section 3.18.69
1BA20h	PW_RX_CAPT0	RIO port-write reception capture 0-3 CSR	Section 3.18.70
1BA24h	PW_RX_CAPT1	RIO port-write reception capture 0-3 CSR	Section 3.18.70
1BA28h	PW_RX_CAPT2	RIO port-write reception capture 0-3 CSR	Section 3.18.70
1BA2Ch	PW_RX_CAPT3	RIO port-write reception capture 0-3 CSR	Section 3.18.70
1BD00h	LLM_BH	IDT-specific LLM block header	Section 3.18.71
1BD24h	WHITEBOARD	RIO Whiteboard CSR	Section 3.18.72
1BD28h	PORT_NUMBER	RIO Port Number CSR	Section 3.18.73
1BD30h	PRESCALAR_SRV_CLK	RIO port IP prescalar	Section 3.18.74
1BD34h	REG_RST_CTL	RIO register reset control CSR	Section 3.18.75
1BD48h	LOCAL_ERR_DET	Local Logical/Transport Layer Error Detect CSR	Section 3.18.76

Table 3-2. SRIO Registers (continued)

Offset	Acronym	Register Description	Section
1BD4Ch	LOCAL_ERR_EN	Local Logical/Transport Layer Error Enable CSR	Section 3.18.77
1BD50h	LOCAL_H_ADDR_CAPT	Local Logical/Transport Layer High Address Capture CSR	Section 3.18.78
1BD54h	LOCAL_ADDR_CAPT	Local Logical/Transport Layer Address Capture CSR	Section 3.18.79
1BD58h	LOCAL_ID_CAPT	Local Logical/Transport Layer DeviceID Capture CSR	Section 3.18.80
1BD5Ch	LOCAL_CTRL_CAPT	Local Logical/Transport Layer Layer Control Capture CSR	Section 3.18.81
1BE00h	FABRIC_HDR	IDT-specific fabric module block header	Section 3.18.82
1BE10h	FABRIC_CSR	Fabric control and status register	Section 3.18.83
1BE40h	SP0_FABRIC_STATUS	RIO port 0-3 fabric status register	Section 3.18.84
1BE44h	SP1_FABRIC_STATUS	RIO port 0-3 fabric status register	Section 3.18.84
1BE48h	SP2_FABRIC_STATUS	RIO port 0-3 fabric status register	Section 3.18.84
1BE4Ch	SP3_FABRIC_STATUS	RIO port 0-3 fabric status register	Section 3.18.84

3.2 SerDes Macro Registers

3.2.1 SerDes Macro Status Register

Figure 3-1. SerDes Macro Status Register (SRIO_SERDES_STS — 0x02620154)

31	29	28	27	26	25	24	23	22
Reserved		TX TESTFAIL	EQOVER	EQUNDER	OCIP	LOSTDTCT	SYNC	RX TESTFAIL
R		STS3 (R)						

21	20	19	18	17	16	15	14	13	12	11
TX TESTFAIL	EQOVER	EQUNDER	OCIP	LOSTDTCT	SYNC	RX TESTFAIL	TX TESTFAIL	EQOVER	EQUNDER	OCIP
STS2 (R)							STS1 (R)			

10	9	8	7	6	5	4	3	2	1	0
LOSTDTCT	SYNC	RX TESTFAIL	TX TESTFAIL	EQOVER	EQUNDER	OCIP	LOSTDTCT	SYNC	RX TESTFAIL	LOCK
STS1 (R)			STS0 (R)							R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate; STSi, STS lane number — see the device-specific data manual

Table 3-3. SerDes Macro Status Register (SRIO_SERDES_STS — 0x02620154) Field Description

Bit	Field	Description
31-29	Reserved	Reserved.
28	TX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
27	EQOVER	*Received signal over equalized.* Driven high asynchronously during equalizer analysis if the received signal is over equalized.
26	EQUNDER	*Received signal under equalized.* Driven high asynchronously during equalizer analysis if the received signal is under equalized.
25	OCIP	*Offset compensation in progress.* Driven high asynchronously during offset compensation.
24	LOSTDTCT	*Loss of Signal detect.* Driven high asynchronously when a loss of signal condition is detected for channel i.
23	SYNC	*Symbol alignment.* When comma detection is enabled, this output is high when an aligned comma is received, in the same cycle that the comma pattern is output on **rd**i. Alternatively, when an alignment jog is requested, it is high to indicate that the request has been completed. Synchronous to **rxbclk**[i].
22	RX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
21	TX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
20	EQOVER	*Received signal over equalized.* Driven high asynchronously during equalizer analysis if the received signal is over equalized.
19	EQUNDER	*Received signal under equalized.* Driven high asynchronously during equalizer analysis if the received signal is under equalized.
18	OCIP	*Offset compensation in progress.* Driven high asynchronously during offset compensation.
17	LOSTDTCT	*Loss of Signal detect.* Driven high asynchronously when a loss of signal condition is detected for channel i.
16	SYNC	*Symbol alignment.* When comma detection is enabled, this output is high when an aligned comma is received, in the same cycle that the comma pattern is output on **rd**i. Alternatively, when an alignment jog is requested, it is high to indicate that the request has been completed. Synchronous to **rxbclk**[i].
15	RX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
14	TX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
13	EQOVER	*Received signal over equalized.* Driven high asynchronously during equalizer analysis if the received signal is over equalized.
12	EQUNDER	*Received signal under equalized.* Driven high asynchronously during equalizer analysis if the received signal is under equalized.
11	OCIP	*Offset compensation in progress.* Driven high asynchronously during offset compensation.

Table 3-3. SerDes Macro Status Register (SRIO_SERDES_STS — 0x02620154) Field Description (continued)

Bit	Field	Description
10	LOSTDTCT	*Loss of Signal detect.* Driven high asynchronously when a loss of signal condition is detected for channel i.
9	SYNC	*Symbol alignment.* When comma detection is enabled, this output is high when an aligned comma is received, in the same cycle that the comma pattern is output on **rd**i. Alternatively, when an alignment jog is requested, it is high to indicate that the request has been completed. Synchronous to **rxbclk**[i].
8	RX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
7	TX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
6	EQOVER	*Received signal over equalized.* Driven high asynchronously during equalizer analysis if the received signal is over equalized.
5	EQUNDER	*Received signal under equalized.* Driven high asynchronously during equalizer analysis if the received signal is under equalized.
4	OCIP	*Offset compensation in progress.* Driven high asynchronously during offset compensation.
3	LOSTDTCT	*Loss of Signal detect.* Driven high asynchronously when a loss of signal condition is detected for channel i.
2	SYNC	*Symbol alignment.* When comma detection is enabled, this output is high when an aligned comma is received, in the same cycle that the comma pattern is output on **rd**i. Alternatively, when an alignment jog is requested, it is high to indicate that the request has been completed. Synchronous to **rxbclk**[i].
1	RX TESTFAIL	*Test failure.* Driven high when an error is encountered during a test sequence executed on channel i. Synchronous to **rxbclk**[i].
0	LOCK	*PLL lock.* Driven high asynchronously between 2048- 3071 **refclkp/n** cycles after the PLL has locked, which will occur within 200 **refclkp/n** cycles. Whilst **LOCK** is low, the PLL output frequency may overshoot by no more than <5%, provided **MPY** is not changed to select a lower multiplication factor. The same percentage overshoot will be mirrored by the bus clocks originating from the SerDes.

3.2.2 SerDes Macro Configuration Register

Figure 3-2. SerDes Macro Configuration Register (SRIO_SERDES_CFGPLL — 0x02620360)

31 15	14 13	12 11	10	9	8 1	0
Reserved	CLKBYP	LOOP BANDWIDTH	SLEEPPLL	VRANGE	MPY	ENPLL
R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-4. SerDes Macro Configuration Register (SRIO_SERDES_CFGPLL — 0x02620360) Field Description

Bit	Field	Description
31-15	Reserved	Reserved.
14-13	CLKBYP	*Clock Bypass*. Facilitates bypassing of PLL with either **refclkp/n** or **testclkt**, and bypassing of the recovered receiver clock with **testclkr**. A full description of the value selection can be found in Table 3-8.
12-11	LOOP BANDWIDTH	Loop bandwidth. Specify loop bandwidth settings. Jitter on the reference clock degrades both the transmit eye and receiver jitter tolerance, thereby impairing system performance. Performance of the integrated PLL is optimized according to the jitter characteristics of the reference clock through the LOOP BANDWIDTH field. 00b = Frequency dependent bandwidth. The PLL bandwidth is set to 1/12 of the RIOCLK/ $\overline{\text{RIOCLK}}$ frequency. This setting is suitable for most systems that input the reference clock through a low jitter input cell, and is required for standard compliance. 01b = Reserved. 10b = Low bandwidth. The PLL bandwidth is set to 1/20 of the RIOCLK/ $\overline{\text{RIOCLK}}$ frequency or 3 MHz, whichever is larger. In systems where the reference clock is directly input a low-jitter input cell, but is of lower quality, this setting may offer better performance. It reduces the amount of reference clock jitter transferred through the PLL. However, it also increases the susceptibility to loop ground noise generated within the PLL itself. It is difficult to predict whether the improvement in the former offsets the degradation in the latter. 11b = High bandwidth. The PLL bandwidth is set to 1/8 of the RIOCLK/ $\overline{\text{RIOCLK}}$ frequency. This is the setting appropriate for systems where the reference clock is cleaned through an ultra-low jitter LC-based PLL. Standards compliance is achieved even if the reference clock input to the cleaner PLL is outside the specification for the standard.
10	SLEEPPLL	*Sleep PLL*. Puts the PLL into the sleep state when high.
9	VRANGE	*VCO range*. Selects between high- and low-range VCO. 0b = Set the VCO speed range to a high frequency. 1b = Set the VCO speed range to a low frequency.
8-1	MPY	*PLL multiply*. Select PLL multiply factors between 4 and 60. 00010000b = 4x 00010100b = 5x 00110000b = 6x 00100000b = 8x 00100001b = 8.25x 00101000b = 10x 00110000b = 12x 00110010b = 12.5x 00111100b = 15x 01000000b = 16x 01000001b = 16.5x 01010000b = 20x 01011000b = 22x 01100100b = 25x
0	ENPLL	*Enable PLL*. • 0h = PLL Disabled • 1h = PLL Enabled

Based on the MPY value, the line rate versus PLL output clock frequency can be calculated. This is summarized in Table 3-5.

Table 3-5. Line Rate Versus PLL Output Clock Frequency

Rate	Line Rate	PLL Output Frequency	RATESCALE
Full	x Gbps	0.25x GHz	0.25
Half	x Gbps	0.5x GHz	0.5
Quarter	x Gbps	1x GHz	1
Eighth	x Gbps	2x GHz	2

The rate is defined by the RATE bits of the SERDES_CFGRX *n*_CNTL register and the SERDES_CFGTX *n*_CNTL register, respectively. The primary operating frequency of the SerDes macro is determined by the reference clock frequency and PLL multiplication factor. However, to support lower frequency applications, each receiver and transmitter can also be configured to operate at a half or quarter of this rate through the RATE bits of the SERDES_CFGRX *n*_CNTL and SERDES_CFGTX *n*_CNTL registers, as described in Table 3-6.

Table 3-6. Effect of the RATE Bits

0 0	Full Rate	Four data samples taken per PLL output clock cycle
0 1	Half Rate	Two data sample taken per PLL output clock cycle
1 0	Quarter Rate	One data sample taken per PLL output clock cycle
1 1	Eighth Rate	One data sample taken every two PLL output clock cycles

Table 3-7 shows the frequency range versus the multiplication factor (MPY).

Table 3-7. Frequency Range versus MPY Value

RefClk (MHz)	MPY	Data Rate (Gbps)			
		Full (0b00)	Half (0b01)	Qtr (0b10)	Eighth (0b11)
156.25	10 (0b00101000)	x	3.125	x	x
156.25	16 (0b01000000)	x	5	2.5	1.25
250	10 (0b00101000)	x	5	2.5	1.25
250	5 (0b00010100)	5	2.5	1.25	x
312.5	5 (0b00010100)	x	3.125	x	x
312.5	8 (0b00100000)	x	5	2.5	1.25

Table 3-8 lists clock bypass selection.

Table 3-8. Clock Bypass

Value	Effect
0 0	*No bypass*. Macro operates normally from the PLL.
0 1	*Reserved*.
1 0	*Functional bypass*. The macro operates functionally at low speed using **testclkt** and **testclkr**.
1 1	*Refclk Observe*. The PLL is bypassed by r **efclkp/n**. This diagnostic capability is designed to facilitate observation of the reference clock from macros containing transmitters.

3.3 SerDes Receive/Transmit Channel Configuration Registers

3.3.1 SerDes Receive Channel Configuration Register[0-3]

Figure 3-3. SerDes Receive Channel Configuration Register n (SRIO_SERDES_CFGRX[0-3]) (0x02620364 +(n * 0x8))

31–28	27–25	24–23	22	21	20–18
Reserved	TESTPATTERN	LOOPBACK	ENOC	EQHLD	EQ
R/W	R/W	R/W	R/W	R/W	R/W

17–15	14–12	11–10	9–7	6–5	4–3	2–1	0
CDR	LOS	ALIGN	TERM	INVPAIR	RATE	BUSWIDTH	ENRX
R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-9. SerDes Receive Channel Configuration Register (SRIO_SERDES_CFGRX[0-3]) Field Descriptions

Bit	Field	Description
31-28	Reserved	Reserved.
27-25	TESTPATTERN	*Enables and selects test patterns*. Enables and selects verification of one of three PRBS patterns, a user-defined pattern or a clock test pattern. 000b = Test mode disabled. 001b = *Alternating 0/1 Pattern*. An alternating 0/1 pattern with a period of 2UI. 010b = *Generate or Verify 2^7 - 1 PRBS.* Uses a 7-bit LFSR with feedback polynomial $x^7 + x^6 + 1$. 011b = *Generate or Verify 2^{23} - 1 PRBS*. Uses an ITU O.150 conformant 23-bit LFSR with feedback polynomial $x^{23} + x^{18} + 1$. 100b = *Generate or Verify 2^{31} - 1 PRBS.* Uses an ITU O.150 conformant 31-bit LFSR with feedback polynomial $x^{31} + x^{28} + 1$. 101b = *User-defined 20-bit pattern.* Uses the USR_PATT IEEE1500 Tuning instruction field to specify the pattern. The default value is 0x66666. 11xb = Reserved.
24-23	LOOPBACK	*Loopback*. Enables loopback, see Table 3-11.
22	ENOC	*Enable offset compensation*. Enables samplers offset compensation.
21	EQHLD	*Hold equalizer*. Holds the equalizer in its current state. 0b = *Equalizer adaption enabled*. The equalizer adaption and analysis algorithm is enabled. This should be the default state. 1b = *Equalizer adaption held.* The equalizer is held in its current state; the adaption and analysis algorithm is reset.
20-18	EQ	000b - 111b = *Equalizer*. Enables and configures the adaptive equalizer to compensate for loss in the transmission media. For the selectable values, see Table 3-10.
17-15	CDR	*Clock/data recovery*. Configures the clock/data recovery algorithm. 000b = Second order. Phase offset tracking up to ±313 ppm with 15-vote threshold. 001b = Second order. Highest precision frequency offset matching, but poorest response to changes in frequency offset, and longest lock time. Suitable for use in systems with fixed frequency offset. 010b = Second order. Medium precision frequency offset matching, frequency offset change response, and lock time. 011b = Second order. Best response to changes in frequency offset and fastest lock time, but lowest precision frequency offset matching. Suitable for use in systems with spread spectrum clocking. 100b = First order with fast lock. Phase offset tracking up to ±1953 ppm in the presence of ...10101010... training pattern and ±868 ppm, otherwise. 101b = First order with fast lock. As per setting 001, but with improved response to changes in frequency offset when not close to lock. 110b = First order with fast lock. As per setting 010, but with improved response to changes in frequency offset when not close to lock. 111b = First order with fast lock. As per setting 011, but with improved response to changes in frequency offset when not close to lock.

Table 3-9. SerDes Receive Channel Configuration Register (SRIO_SERDES_CFGRX[0-3]) Field Descriptions (continued)

Bit	Field	Description
14-12	LOS	*Loss of signal*. Enables loss of signal detection with two selectable thresholds. 000b = Disabled. Loss of signal detection disabled. 001b-01xb = High threshold. Loss of signal detection threshold in the range of 85 to 195 mV_{dftt}. This setting is suitable for infiniband. 100b = Low threshold. Loss of signal detection threshold in the range of 65 to 175 mV_{dftt}. This setting is suitable for PCI-E and S-ATA. 101b-11xb = Reserved
11-10	ALIGN	*Symbol alignment*. Enables internal or external symbol alignment. 00b = Alignment disabled. No symbol alignment will be performed while this setting is selected, or when switching to this selection from another. 01b = Comma alignment enabled. Symbol alignment will be performed whenever a misaligned comma symbol is received. 10b = Alignment jog. The symbol alignment will be adjusted by one bit position when this mode is selected (that is, the ALIGN value changes from 0xb to 1xb). 11b = Reserved
9-7	TERM	*Input termination*. The only valid value for this field is 001b; all other values are reserved. The value 001b sets the common point to 0.8 VDDT and supports AC-coupled systems using CML transmitters. The transmitter has no effect on the receiver common mode, which is set to optimize the input sensitivity of the receiver. Common-mode termination is through a 50-pF capacitor to VSSA.
6	INVPAIR	*Invert polarity*. Inverts polarity of RIORXn and $\overline{\text{RIORXn}}$. 0h = Normal polarity. RIORXn is considered to be positive data and $\overline{\text{RIORXn}}$ negative. 1h = Inverted polarity. RIORXn is considered to be negative data and $\overline{\text{RIORXn}}$ positive.
5-4	RATE	Operating rate. Selects full, half, quarter, or eighth rate operation. 00b = Full rate. Four data samples taken per PLL output clock cycle. 01b = Half rate. Two data samples taken per PLL output clock cycle. 10b = Quarter rate. One data sample taken per PLL output clock cycle. 11b = Eighth rate. One data sample taken every two PLL output clock cycles.
3-1	BUSWIDTH	*Bus width*. Always write 010b to this field to indicate a 20-bit wide parallel bus to the clock. All other values are reserved. See Section 2.3.3.1 for an explanation of the bus.
0	ENRX	*Enable receiver*. 0h = Disable this receiver. 1h = Enable this receiver.

Table 3-10. EQ Bits

Value	Effect
EQ[1:0] — SRIO_SERDES_CFGRX(n)[19-18]	
0 0	No equalization. The equalizer provides a flat response at the maximum gain. The setting may be appropriate if jitter at the receiver occurs predominantly as a result of crosstalk rather than frequency-dependent loss.
0 1	Fully adaptive equalization. The zero position is determined by the selected operating rate, and the low-frequency gain of the equalizer is determined algorithmically by analyzing the data patterns and transition positions in the received data. This setting should be used for most applications.
1 0	Precursor equalization analysis. The data patterns and transition positions in the received data are analyzed to determine whether the transmit link partner is applying more or less precursor equalization than necessary. See the *SerDes Implementation Guidelines for KeyStone I Devices* (SPRABC1) application note for more information.
1 1	Postcursor equalization analysis. The data patterns and transition positions in the received data are analyzed to determine whether the transmit link partner is applying more or less precursor equalization than necessary. See the *SerDes Implementation Guidelines for KeyStone I Devices* (SPRABC1) application note for more information.
EQ[2] — SRIO_SERDES_CFGRX(n)[20]	
0	Default
1	Boost. Equalizer gain is boosted by 6 dB, with a 20 percent reduction in bandwidth, and an increase of 5-mW power consumption. May improve performance over long links. Should be selected only in applications using 1 V vddt.

The loopback is described below:

Table 3-11. Loopback—cfgrxi[24-23]

Value	RX Effect
0 0	*Disabled*.
0 1	Reserved.
1 0	Reserved.
1 1	*Loopback*. The differential current from the transmitter is converted to a voltage and applied to the receiver input if LOS (cfgrx *i*[15-13]) = 0 0, or to the Loss of signal detector if LOS = 10.

3.3.2 SerDes Transmit Channel Configuration Register[0-3]

Figure 3-4. SerDes Transmit Channel Configuration Register n (SERDES_CFGTXn_CNTL) (0x02620368 + (n * 0x8))

31–26	25–23	22–21	20	19	18–14
Reserved	TESTPATTERN	LOOPBACK	MSYNC	FIRUPT	TWPST1
R/W	R/W	R/W	R/W	R/W	R/W

13–11	10–7	6–5	4–3	1	0
TWPRE	SWING	INVPAIR	RATE	BUSWIDTH	ENTX
R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-12. SerDes Transmit Channel Configuration Register n (SERDES_CFGTXn_CNTL) Field Descriptions

Bit	Field	Description
31-26	Reserved	Reserved.
25-23	TESTPATTERN	*Enables and selects test patterns*. Enables and selects verification of one of three PRBS patterns, a user-defined pattern or a clock test pattern. 000b = Test mode disabled. 001b = *Alternating 0/1 Pattern*. An alternating 0/1 pattern with a period of 2UI. 010b = *Generate or Verify 2^7 - 1 PRBS.* Uses a 7-bit LFSR with feedback polynomial $x^7 + x^6 + 1$. 011b = *Generate or Verify 2^{23} - 1 PRBS*. Uses an ITU O.150 conformant 23-bit LFSR with feedback polynomial $x^{23} + x^{18} + 1$. 100b = *Generate or Verify 2^{31} - 1 PRBS.* Uses an ITU O.150 conformant 31-bit LFSR with feedback polynomial $x^{31} + x^{28} + 1$. 101b = *User-defined 20-bit pattern.* Uses the USR_PATT IEEE1500 Tuning instruction field to specify the pattern. The default value is 0x66666. 11xb = Reserved.
22-21	LOOPBACK	*Loopback*. Enables loopback.
20	MSYNC	1h = *Synchronization master*. Enables the channel as the master lane for synchronization purposes. Tie low only for slave lanes in aggregated links. For single-lane applications, tie MSYNC high. When aggregating multiple lanes, precisely one lane per aggregate must have MSYNC set to 1, and it must always be the lowest numbered channel in the aggregate.
19	FIRUPT	0h = *Transmitter pre and post cursor FIR filter update*. Update control of FIR tap weights. fields **TWPRE** and **TWPST1** can be updated when SerDes byte clock and this input are both high.
18-14	TWPST1	00000b-11111b = *Adjacent post cursor Tap weight*. Selects one of 32 output tap weights for TX waveform conditioning. The settings range from -37.5 to +37.5% in 2.5% steps.
13-11	TWPRE	000b-111b = *Precursor Tap weight*. Selects one of 8 output tap weights for TX waveform conditioning. The settings range from 0 to -17.5% in 2.5% steps.
10-7	SWING	0000b-1111b = *Output swing*. Selects one of 16 output amplitude settings between 795 and 1275 mV_{dfpp}. See Table 3-13.
6	INVPAIR	*Invert polarity*. Inverts polarity of RIOTXn and $\overline{\text{RIOTXn}}$. 0h = Normal polarity. RIOTXn is considered to be positive data and $\overline{\text{RIOTXn}}$ negative. 1h = Inverted polarity. RIOTXn is considered to be negative data and $\overline{\text{RIOTXn}}$ positive.
5-4	RATE	*Operating rate.* Selects full, half, quarter, or eighth rate operation. 00b = Full rate. Four data samples taken per PLL output clock cycle. 01b = Half rate. Two data samples taken per PLL output clock cycle. 10b = Quarter rate. One data sample taken per PLL output clock cycle. 11b = Eighth. One data samples taken every two PLL output clock cycles.
3-1	BUSWIDTH	*Bus width*. Always write 010b to this field to indicate a 20-bit wide parallel bus to the clock. All other values are reserved. See Section 2.3.3.1 for an explanation of the bus.
0	ENTX	*Enable transmitter*. 0h = Disable this transmitter. 1h = Enable this transmitter.

Table 3-13. SWING Bits—cfgtxi [10-7]

Value	Amplitude (mV_{dfpp})	Common Mode (mV)
0 0 0 0	795	440
0 0 0 1	830	460
0 0 1 0	870	480
0 0 1 1	905	500
0 1 0 0	940	525
0 1 0 1	975	545
0 1 1 0	1010	560
0 1 1 1	1045	580
1 0 0 0	1080	600
1 0 0 1	1110	620
1 0 1 0	1145	635
1 0 1 1	1175	655
1 1 0 0	1200	670
1 1 0 1	1230	690
1 1 1 0	1255	700
1 1 1 1	1275	715

Table 3-14. Pre-cursor Transmit Tap Weights—cfgtxi [13-11]

Value	Tap Weight (%)
0 0 0	0
0 0 1	–2.5
0 1 0	–5.0
0 1 1	–7.5
1 0 0	–10.0
1 0 1	–12.5
1 1 0	–15.0
1 1 1	–17.5

Table 3-15. Post-cursor Transmit Tap Weights—cfgtxi [18-14]

Value	Positive Tap Weight (%)	Value	Negative Tap Weight (%)
0 0 0 0 0	0	1 0 0 0 0	0
0 0 0 0 1	+2.5	1 0 0 0 1	–2.5
0 0 0 1 0	+5.0	1 0 0 1 0	–5.0
0 0 0 1 1	+7.5	1 0 0 1 1	–7.5
0 0 1 0 0	+10.0	1 0 1 0 0	–10.0
0 0 1 0 1	+12.5	1 0 1 0 1	–12.5
0 0 1 1 0	+15.0	1 0 1 1 0	–15.0
0 0 1 1 1	+17.5	1 0 1 1 1	–17.5
0 1 0 0 0	+20.0	1 1 0 0 0	–20.0
0 1 0 0 1	+22.5	1 1 0 0 1	–22.5
0 1 0 1 0	+25.0	1 1 0 1 0	–25.0
0 1 0 1 1	+27.5	1 1 0 1 1	–27.5
0 1 1 0 0	+30.0	1 1 1 0 0	–30.0
0 1 1 0 1	+32.5	1 1 1 0 1	–32.5
0 1 1 1 0	+35.0	1 1 1 1 0	–35.0
0 1 1 1 1	+37.5	1 1 1 1 1	–37.5

Table 3-16. Loopback—cfgtxi [22-21]

Value	TX Effect
0 0	*Disabled.*
0 1	Reserved.
1 0	*Loopback TX driver disabled.* The loopback path covers all the stages of the transmitter except the TX output itself. A differential current is passed to the receiver. The magnitude of this current is dependent on **SWING (cfgtxi[12-9])**. THe transmit driver itself is disabled.
1 1	*Loopback TC driver enabled.* As above, but the transmit driver operates normally.

3.4 Required Peripheral Registers

3.4.1 Peripheral ID Register (PID)

The peripheral identification register (PID) is a constant register that contains the ID and ID revision number for that peripheral. The PID stores version information used to identify the peripheral. All bits within this register are read-only (writes have no effect), meaning that the values within this register should be hard-coded with the appropriate values and must not change from their reset state.

Figure 3-5. Peripheral ID Register (Address Offset 0x0000)

31 30	29 28	27 16	15 11	10 8	7 6	5 0
Scheme	BU	Func	RTL	Major	Custom	Minor
R						

Legend: R = Read only; W = Write only

Table 3-17. Peripheral ID Register (Address Offset 0x0000)

Bit	Name	Access	Reset Value	Description
31-30	Scheme	R	2'b01	PID Scheme
29-28	BU	R	2'b00	Business Unit
27-16	Func	R	-	Indicates software compatibility module. Assigned by IP Team
15-11	RTL	R	-	This field indicates a major release of the IP. This revision information changes based on the release being made.
10-8	Major	R	-	This field indicates a major release of the IP. This revision information changes based on the release being made.
7-6	Custom	R		This field indicates a minor release of the IP. This revision information changes based on the release being made
5-0	Minor	R	-	This field indicates a minor release of the IP. This revision information changes based on the release being made

3.4.2 Peripheral Control Register

The peripheral control register (PCR) contains a bit that enables or disables the entire peripheral, and one bit for every module within the peripheral where this level of control is desired. The module control bits can only be written when the peripheral itself is enabled. In addition, the PCR has emulation control bits free and soft, which control the peripheral behavior during emulation halts.

Figure 3-6. Peripheral Control Register (PCR) (Address offset 0x0004)

31	5	4	3	2	1	0
Reserved		Restore	Local_Dis	PEREN	SOFT	FREE
		R/W	R	R/W	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-18. Peripheral Control Register (Address offset 0x0004)

Bit	Name	Access	Reset Source	Reset Value	Description
31-5	Reserved	-	VBUS_rst	0	Reserved
4	Restore	R/W	VBUS_reset	0b	Writing 1 to this bit clears the Local_Dis bit and allows for reception of local packets.
3	Local_DIS	R	VBUS_reset	0b	Local Disable Indicates discarding of incoming packets during CPU reset, while allowing packet forwarding to continue operating. This is controlled by the Rod_mod_g_rst_n signal coming into RapidIO. • 0b0 — Local RX traffic is discarded • 0b1 — Local RX traffic is accepted normally
2	PEREN	R/W	VBUS_rst	0	Peripheral Enable — Controls the flow of data in the logical layer of the peripheral. As an initiator, it prevents TX transaction generation; as a target, it disables incoming requests. This should be the last enable bit to toggle when bringing the device out of reset to begin normal operation. • 0b0 - disabled • 0b1 - enabled
1	SOFT	R/W	VBUS_rst	0	Emulation control — SOFT bit
0	FREE	R/W	VBUS_rst	1	Emulation control — FREE bit

3.5 Peripheral Setting Control Registers

3.5.1 Peripheral Setting Control Register (PER_SET_CNTL)

Table 3-19. Peripheral Settings Control Register (RIO_PER_SET_CNTL)(Address offset 0x0014)

31 30	29 28	27	26 25
MAU_LEND_SWAP_MODE	LSU_LEND_SWAP_MODE	LOG_TGT_ID_DIS	Reserved
R/W	R/W	R/W	R/W

spacer

24	23 22	21	20 18	17 15	14 12	11 9
BOOT _COMPLETE	TXU/RXU_LEND_SWAP_MODE	PROMOTE _DIS	TX_PRI2 _WM	TX_PRI1 _WM	TX_PRI0 _WM	CBA_TRAN _PRI
R/W	R/W	R/W	R/W	R/W	R/W	R/W

spacer

8	7	6 3	2	1	0
Reserved	PRESCALER _SELECT	SERDES3_PRBS _OVR	SERDES2_PRBS _OVR	SERDES1_PRBS _OVR	SERDES0_PRBS _OVR
R	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-20. Peripheral Settings Control Register (Address offset 0x0014)

Bit	Name	R/W	Reset Value (vbusp_rst)	Description
31-30	MAU_LEND_SWAP_MODE	R/W	0b00	MAU little-endian swapping mode: • 0b00 — Mode A (1Byte) • 0b01 — Mode B (2Byte) • 0b10 — Mode C (4Byte) • 0b11 — Mode D (8Byte)
29-28	LSU_LEND_SWAP_MODE	R/W	0b00	LSU little-endian swapping mode. • 0b00 — Mode A (1Byte) • 0b01 — Mode B (2Byte) • 0b10 — Mode C (4Byte) • 0b11 — Mode D (8Byte)
27	LOG_TGT_ID_DIS	R/W	0b0	0b0 — Supports only packets where DestID equals BASE_ID or any of the device IDs or 8 multicast IDs. 0b1 — Supports promiscuous ID/Unicast IDs. All packets regardless of DestID are accepted by the logical layer and handled by the appropriate functional block.
26-25	RESERVED	R/W	0b00	Reserved
24	BOOT_COMPLETE	R/W	0b0	Controls ability to write any register during initialization. It also includes read only registers during normal mode of operation, that have application defined reset value. 0 - write enabled 1 - write to read only registers disabled The boot_complete is asserted once after reset to define power on configuration. Physical layer initialization process is started once boot_complete is set.
23-22	TXU/RXU_LEND_SWAP _MODE	R/W	0b00	TXU/RXUlittle-endian swapping mode: • 0b00 — Mode A (1Byte) • 0b01 — Mode B (2Byte) • 0b10 — Mode C (4Byte) • 0b11 — Mode D (8Byte)

Table 3-20. Peripheral Settings Control Register (Address offset 0x0014) (continued)

Bit	Name	R/W	Reset Value (vbusp_rst)	Description
21	PROMOTE_DIS	R/W	0b0	Disable automatic promotion of response priority by RXU and MAU. 0b0 — normal 0b1 — Disable auto promotion, response is only be sent back at request+1 priority level.
20-18	TX_PRI2_WM	R/W	0b001	Transmit credit threshold. Sets the required number of logical layer TX buffers needed to send priority 2 packets across the UDI interface. (Valid for all ports in all modes) • TX_PRI2_WM Required Buffer Count • 000 => 8, 7, 6, 5, 4, 3, 2, 1 • 001 => 8, 7, 6, 5, 4, 3, 2 • 010 => 8, 7, 6, 5, 4, 3 • 011 => 8, 7, 6, 5, 4 • 100 => 8, 7, 6, 5 • 101 => 8, 7, 6 • 110 => 8, 7 • 111 => 8
17-15	TX_PRI1_WM	R/W	0b010	Transmit credit threshold. Sets the required number of logical layer TX buffers needed to send priority 1 packets across the UDI interface. (Valid for all ports in all modes) TX_PRI1_WM Required Buffer Count • 000 => 8, 7, 6, 5, 4, 3, 2, 1 • 001 => 8, 7, 6, 5, 4, 3, 2 • 010 => 8, 7, 6, 5, 4, 3 • 011 => 8, 7, 6, 5, 4 • 100 => 8, 7, 6, 5 • 101 => 8, 7, 6 • 110 => 8, 7 • 111 => 8
14-12	TX_PRI0_WM	R/W	0b011	Transmit credit threshold. Sets the required number of logical layer TX buffers needed to send priority 0 packets across the UDI interface. (Valid for all ports in all modes) • TX_PRI0_WM Required Buffer Count • 000 => 8, 7, 6, 5, 4, 3, 2, 1 • 001 => 8, 7, 6, 5, 4, 3, 2 • 010 => 8, 7, 6, 5, 4, 3 • 011 => 8, 7, 6, 5, 4 • 100 => 8, 7, 6, 5 • 101 => 8, 7, 6 • 110 => 8, 7 • 111 => 8
11-9	CBA_TRANS_PRI	R/W	0b100	VBUS transaction priority 0b000 — Highest Priority … 0b111 — Lowest Priority
8	Reserved	RO	0b0	Reserved

Table 3-20. Peripheral Settings Control Register (Address offset 0x0014) (continued)

Bit	Name	R/W	Reset Value (vbusp_rst)	Description
7-4	PRESCALER_SELECT	R/W	0b0000	VBUS frequency prescaler, used to drive the request-to-response timers. These 4 bits are the prescaler reload value allowing division of the VBUS clock by a range from 1 up to 16. Setting should reflect the device VBUS frequency. Prescale Min VBUS Max VBUS • Bits Freq(MHz) Freq(MHz) • 0000 44.7 89.5 • 0001 89.5 179.0 • 0010 134.2 268.4 • 0011 180.0 360.0 • 0100 223.7 447.4 • 0101 268.4 536.8 • 0110 313.2 626.4 • 0111 357.9 715.8 • 1000 402.7 805.4 • 1001 447.4 894.8 • 1010 492.1 984.2 • 1011 536.9 1073.8 • 1100 581.6 1163.2 • 1101 626.3 1252.6 • 1110 671.1 1342.2 • 1111 715.8 1431.6
3	SerDes3_PRBS_Ovr	R/W	0b0	SerDes PRBS over-ride disables the physical layer ENTX3, ENRX3 control so that the SerDes can be programmed to generate a PRBS signal. 0b0 — normal 0b1 — Over-ride for PRBS
2	SerDes2_PRBS_Ovr	R/W	0b0	SerDes PRBS over-ride disables the physical layer ENTX2, ENRX2 control so that the SerDes can be programmed to generate a PRBS signal. 0b0 — normal 0b1 — Over-ride for PRBS
1	SerDes1_PRBS_Ovr	R/W	0b0	SerDes PRBS over-ride disables the physical layer ENTX1, ENRX1 control so that the SerDes can be programmed to generate a PRBS signal. 0b0 — normal 0b1 — Over-ride for PRBS
0	SerDes0_PRBS_Ovr	R/W	0b0	SerDes PRBS over-ride disables the physical layer ENTX0, ENRX0 control so that the SerDes can be programmed to generate a PRBS signal. 0b0 — normal 0b1 — Over-ride for PRBS

3.5.2 Peripheral Settings Control Register 1

Figure 3-7. PER_SET_CNTL1 (Address offset 0x0018)

31 16	15 12	11 10	9	8 4	3 2	1	0
RSVD	TXU_RETRY_TIMER_MODE	RSVD	SYS_CLK_VBUSP	COS_EN	Loopback	SYS_CLK_SEL	RXU_WATERMARK	CRF
R	RW	R	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-21. PER_SET_CNTL1 (Address offset 0x0018)

Bit	Name	Type	Reset By	Reset State	Function
31-16	RSVD	R	VBUS_RST	0x0	Reserved
15-12	TXU_RETRY_TIMER_MODE	R/W	VBUS_rst	0x0	These bits are only supported on later Keystone devices. They define the total number of times the 14b Retry Timer can expire, before TXU discards the retry request. “000” = 1 iteration of 14 bit counter “001” = 2 iterations “010” = 4 iterations “011” = 8 iterations “100” = 16 iterations “101” = 32 iterations “110”, “111” = reserved
11-10	RSVD	R	VBUS_rst	2’b0	Reserved
9	SYS_CLK_VBUSP	R/W	VBUS_RST	0x0	These bits are only supported on later Keystone devices. These bits are used in conjunction with SYS_CLK_SEL bits defined in the same register. 0 — Used SYS_CLK_SEL to define the clock source SYS_CLK 1 — Use vbusp_clk as the clock source for SYS_CLK (provided txbxlks are slower than vbusp_clk)
8	COS_EN	R/W	VBUS_RST	0x0	Class of Service Enable 0 -> COS is not a part of the segmentation context for the RXU 1 ->COS is a part of the segmentation context for the RXU
7-4	Loopback [3-0]	R/W	VBUS_RST	4’b0	0b0 — Normal operation 0b1 — Loopback Transmit data to receive on the same port. Packet data is looped back in the digital domain before the SerDes macros • Loopback[0] — Loopback lane0 • Loopback[1] - Loopback lane1 • Loopback[2] - Loopback lane2 • Loopback[3] - Loopback lane3
3-2	SYS_CLK_SEL	R/W	VBUS_RST	2’b00	IP_CLK is selected based on: • 00 — TX0 is the source • 01 — TX1 is the source • 10 — TX2 is the source • 11 — TX3 is the source The associated port does not have to be active, however the SerDes must be configured correctly for the master source
1	RXU_WATERMARK (1)	R/W	VBUS_RST	0x0	0 — RXU does not block any context based on the priority. 1 — RXU assigns a new context only if there are • - More than 2 context available • - Two context available and incoming request has Priority 1 or 2 • - One context available and incoming request has Priority 2
0	CRF	R/W	VBUS_RST	0x0	0 — On a response CRF is the same as the incoming request 1- On a response, CRF is always = 1

(1) RXU_WATERMARK is an optional feature and implementation of this is not decided yet.

3.6 Global and Block Enable Registers

3.6.1 Global Enable Registers

Figure 3-8. Global Enable Register (RIO_GBL_EN)

31	1	0
Reserved		EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-22 for details.

Figure 3-9. Global Enable State Register (RIO_GBL_EN_STAT)

31	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		BLK9_EN_STAT	BLK8_EN_STAT	BLK7_EN_STAT	BLK6_EN_STAT	BLK5_EN_STAT	BLK4_EN_STAT	BLK3_EN_STAT	BLK2_EN_STAT	BLK1_EN_STAT	BLK0_EN_STAT	GBL_EN_STAT
R	R	R	R	R	R	R	R	R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-22 for details.

3.6.2 Block Enable Registers (0-9)

Figure 3-10. Block Enable Register (RIO_BLKn_EN)

31	1	0
Reserved		EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-22 for details.

Figure 3-11. Block Enable State Register (RIO_BLKn_EN_STAT)

31	1	0
Reserved		EN_STAT
R		R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-22 for details.

Table 3-22. EN/EN_STAT Bit Field Descriptions

Bit	Name [1]	Access	Reset Source	Default	Description
0	GBL_EN	R/W	VBUS_reset	0b1	Controls reset to all clock domains within the peripheral • 0 = Peripheral to be disabled (held in reset, clocks disabled) • 1 = Peripheral to be enabled
0	GBL_EN_STAT	R	VBUS_reset	0b1	Indicates state of GBL_EN reset signal • 0 = Peripheral in reset and all clocks are off • 1 = Peripheral enabled and clocking.
0	BLK0_EN	R/W	VBUS_reset	0b1	Controls reset to 0th clock/logical domain. By convention, BLK0 should always be the MMR Periph control registers. • 0 = Logical block 0 to be disabled • 1 = Logical block 0 to be enabled
0	BLK0_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK0_EN reset signal • 0 = Logical block 0 disabled • 1 = Logical block 0 enabled
0	BLK1_EN	R/W	VBUS_reset	0b1	Controls reset to Logical block 1. • 0 = Logical block 1 disabled (held in reset, clocks disabled) • 1 = Logical block 1 enabled
0	BLK1_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK1_EN reset signal • 0 = Logical block 1 in reset and clock is off • 1 = Logical block 1 enabled and clocking.
0	BLK2_EN	R/W	VBUS_reset	0b1	Controls reset Logical block 2. • 0 = Logical block 2 disabled (held in reset, clocks disabled) • 1 = Logical block 2 enabled
0	BLK2_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK2_EN reset signal • 0 = Logical block 2 reset and clock is off • 1 = Logical block 2 enabled and clocking.
0	BLK3_EN	R/W	VBUS_reset	0b1	Controls reset Logical block 3. • 0 = Logical block 3 disabled (held in reset, clocks disabled) • 1 = Logical block 3 enabled
0	BLK3_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK3_EN reset signal • 0 = Logical block 3 reset and clock is off • 1 = Logical block 3 enabled and clocking.

(1) BLKn_EN_STAT bitfields are readable both in the BLKn_EN_STAT_REG and the GBL_EN_STAT_REG

Table 3-22. EN/EN_STAT Bit Field Descriptions (continued)

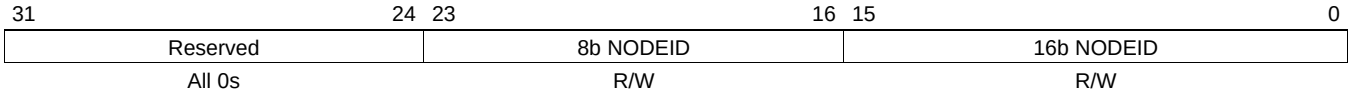
Bit	Name [1]	Access	Reset Source	Default	Description
0	BLK4_EN	R/W	VBUS_reset	0b1	Controls reset Logical block 4. • 0 = Logical block 4 disabled (held in reset, clocks disabled) • 1 = Logical block 4 enabled
0	BLK4_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK4_EN reset signal • 0 = Logical block 4 reset and clock is off • 1 = Logical block 4 enabled and clocking.
0	BLK5_EN	R/W	VBUS_reset	0b1	Controls reset Logical block 5. • 0 = Logical block 5 disabled (held in reset, clocks disabled) • 1 = Logical block 5 enabled
0	BLK5_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK5_EN reset signal • 0 = Logical block 5 reset and clock is off • 1 = Logical block 5 enabled and clocking.
0	BLK6_EN	R/W	VBUS_reset	0b1	Controls reset to Logical block 6. • 0 = Logical block 6 disabled (held in reset, clocks disabled) • 1 = Logical block 6 enabled
0	BLK6_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK6_EN reset signal • 0 = Logical block 6 in reset and clock is off • 1 = Logical block 6 enabled and clocking.
0	BLK7_EN	R/W	VBUS_reset	0b1	Controls reset to Logical block 7. • 0 = Logical block 7 disabled (held in reset, clocks disabled) • 1 = Logical block 7 enabled
0	BLK7_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK7_EN reset signal • 0 = Logical block 7 in reset and clock is off • 1 = Logical block 7 enabled and clocking.
0	BLK8_EN	R/W	VBUS_reset	0b1	Controls reset to Logical block 8. • 0 = Logical block 8 disabled (held in reset, clocks disabled) • 1 = Logical block 8 enabled
0	BLK8_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK8_EN reset signal • 0 = Logical block 8 in reset and clock is off • 1 = Logical block 8 enabled and clocking.
0	BLK9_EN	R/W	VBUS_reset	0b1	Controls reset to Logical block 9. • 0 = Logical block 9 disabled (held in reset, clocks disabled) • 1 = Logical block 9 enabled
0	BLK9_EN_STAT	R	VBUS_reset	0b1	Indicates state of BLK9_EN reset signal • 0 = Logical block 9 in reset and clock is off • 1 = Logical block 9 enabled and clocking.

The GBL_EN register is implemented with a single ENABLE bit. This bit is logically OR'd with the reset input to the module and is fanned out to ALL logical blocks within the peripheral.

3.7 ID Registers

3.7.1 RapidIO MultiID Register 1

Figure 3-12. RapidIO MultiID Register 1



Legend: R = Read only; W = Write only

Table 3-23. RapidIO MultiID Register 1 (Address offset 0x00A0)

Bit	Name	Access	Reset Source	Reset Value	Description
31-24	RSV			All 0s	Reserved
23-16	8b NODEID	R/W	VBUS_rst	FFh	This is a secondary supported DeviceID checked against an in-coming packet's DestID field and used for multicast support.
15-0	16b NODEID	R/W	VBUS_rst	FFFFh	This is a secondary supported DeviceID checked against an in-coming packet's DestID field and used for multicast support.

3.7.2 RapidIO MultiID Register 2–8

Figure 3-13. RapidIO MultiID Register 2 – 8 (Address offset 0x00A4)

31 24	23 16	15 0
Reserved	8b NODEID	16b NODEID
All 0s	R/W	R/W

Legend: R = Read only; W = Write only

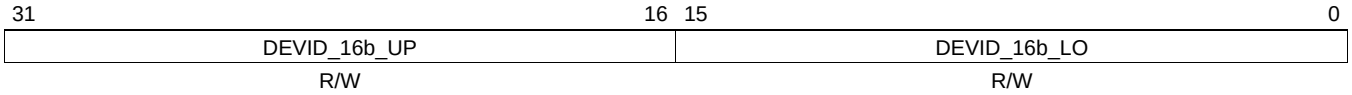
Table 3-24. RapidIO MultiID Register 2 – 8 (Address offset 0x00A4)

Bit	Name	Access	Reset Source	Reset Value	Description
31-24	RSV			All 0s	Reserved
23-16	8b NODEID	R/W	VBUS_rst	FFh	This is a secondary supported DeviceID checked against an in-coming packet's DestID field and used for multicast support.
15-0	16b NODEID	R/W	VBUS_rst	FFFFh	This is a secondary supported DeviceID checked against an in-coming packet's DestID field and used for multicast support.

3.8 Hardware Packet Forwarding Registers

3.8.1 PF_16b_CNTL[0-7]

Figure 3-14. PF_16b_CNTL[0-7]



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-25. PF_16b_CNTL[0-7]

Bit	Control/Command Register Field	R/W	Reset value (vbus_rst)	Description
31-16	DEVID_16b_UP	R/W	16'hFFFF	Upper 16b DeviceID boundary. DestID above this range cannot use the table entry
15-0	DEVID_16b_LO	R/W	16'hFFFF	Lower 16b DeviceID boundary. DestID lower than this number cannot use the table entry

3.8.2 PF_8b_CNTL[0-7]

Figure 3-15. PF_8b_CNTL[0-7]

31 18	17 16	15 8	7 0
Reserved	OUT_PORT	DEVID_8b_UP	DEVID_8b_LO
R	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-26. PF_8b_CNTL[0-7]

Bit	Control/Command Register Field	R/W	Reset value (vbus_rst)	Description
31-18	RSVD	RO	14'b0	Reserved
17-16	OUT_PORT	R/W	2'b11	Output port number for packets whose DestID falls within the 8b or 16b range for this table entry
15-8	DEVID_8b_UP	R/W	8'hFF	Upper 8b DeviceID boundary. DestID above this range cannot use the table entry
7-0	DEVID_8b_LO	R/W	8'hFF	Lower 8b DeviceID boundary. DestID lower than this number cannot use the table entry

3.9 Interrupt Registers

3.9.1 DOORBELLn Interrupt Condition Status Register (DOORBELL[0-3]_ICSR)

Figure 3-16. Doorbell n Interrupt Condition Status Register

31															16
Reserved															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICS15	ICS14	ICS13	ICS12	ICS11	ICS10	ICS9	ICS8	ICS7	ICS6	ICS5	ICS4	ICS3	ICS2	ICS1	ICS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-27. Doorbell n Interrupt Condition Status Register Field Descriptions

Bit	Field	Description
31-16	Reserved	These read-only bits return 0s when read.
15-0	ICSx (x = 15 to 0)	Doorbell n interrupt condition status bit. • 0h = Bit x of the doorbell information value is 0. • 1h = Bit x of the doorbell information value is 1, generating an interrupt request.

3.9.2 DOORBELLn Interrupt Condition Clear Register (DOORBELL[0-3]_ICCR)

Figure 3-17. Doorbell n Interrupt Condition Clear Register

31															16
Reserved															
R															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICC15	ICC14	ICC13	ICC12	ICC11	ICC10	ICC9	ICC8	ICC7	ICC6	ICC5	ICC4	ICC3	ICC2	ICC1	ICC0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-28. Doorbell n Interrupt Condition Clear Register Field Descriptions

Bit	Field	Description
31-16	Reserved	These read-only bits return 0s when read.
15-0	ICCx (x = 15 to 0)	Doorbell n interrupt condition clear bit. • 0h = No effect. • 1h = Clear bit x of the corresponding doorbell interrupt condition status register (ICSR).

3.9.3 LSU0 Interrupt Status and Clear Register

Figure 3-18. Interrupt Condition Status Register (LSU0_ICSR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICS31	ICS30	ICS29	ICS28	ICS27	ICS26	ICS25	ICS24	ICS23	ICS22	ICS21	ICS20	ICS19	ICS18	ICS17	ICS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICS15	ICS14	ICS13	ICS12	ICS11	ICS10	ICS9	ICS8	ICS7	ICS6	ICS5	ICS4	ICS3	ICS2	ICS1	ICS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-19. Interrupt Condition Clear Register (LSU0_ICCR) (Address Offset 0x01A8)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ICC31	ICC30	ICC29	ICC28	ICC27	ICC26	ICC25	ICC24	ICC23	ICC22	ICC21	ICC20	ICC19	ICC18	ICC17	ICC16
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ICC15	ICC14	ICC13	ICC12	ICC11	ICC10	ICC9	ICC8	ICC7	ICC6	ICC5	ICC4	ICC3	ICC2	ICC1	ICC0
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Where:

- ICS0 – SRCID0, Transaction complete, No Errors (Posted/Non-posted)
- ICS1 – SRCID1, Transaction complete, No Errors (Posted/Non-posted)
- ICS2 – SRCID2, Transaction complete, No Errors (Posted/Non-posted)
- ICS3 – SRCID3, Transaction complete, No Errors (Posted/Non-posted)
- ICS4 – SRCID4, Transaction complete, No Errors (Posted/Non-posted)
- ICS5 – SRCID5, Transaction complete, No Errors (Posted/Non-posted)
- ICS6 – SRCID6, Transaction complete, No Errors (Posted/Non-posted)
- ICS7 – SRCID7, Transaction complete, No Errors (Posted/Non-posted)
- ICS8 – SRCID8, Transaction complete, No Errors (Posted/Non-posted)
- ICS9 – SRCID9, Transaction complete, No Errors (Posted/Non-posted)
- ICS10 – SRCID10, Transaction complete, No Errors (Posted/Non-posted)
- ICS11 – SRCID11, Transaction complete, No Errors (Posted/Non-posted)
- ICS12 – SRCID12, Transaction complete, No Errors (Posted/Non-posted)
- ICS13 – SRCID13, Transaction complete, No Errors (Posted/Non-posted)
- ICS14 – SRCID14, Transaction complete, No Errors (Posted/Non-posted)
- ICS15 – SRCID15, Transaction complete, No Errors (Posted/Non-posted)
- ICS16 – SRCID0, Transaction complete with error
- ICS17 – SRCID1, Transaction complete with error
- ICS18 – SRCID2, Transaction complete with error
- ICS19 – SRCID3, Transaction complete with error
- ICS20 – SRCID4, Transaction complete with error
- ICS21 – SRCID5, Transaction complete with error
- ICS22 – SRCID6, Transaction complete with error
- ICS23 – SRCID7, Transaction complete with error
- ICS24 – SRCID8, Transaction complete with error
- ICS25 – SRCID9, Transaction complete with error
- ICS26 – SRCID10, Transaction complete with error

- ICS27 – SRCID11, Transaction complete with error
- ICS28 – SRCID12, Transaction complete with error
- ICS29 – SRCID13, Transaction complete with error
- ICS30 – SRCID14, Transaction complete with error
- ICS31 – SRCID15, Transaction complete with error

NOTE: Enable for these interrupts is ultimately controlled by the Interrupt Req register bit of the Load/Store command registers. This allows enabling and disabling on a per request basis. For optimum LSU performance, interrupt pacing should not be used on the LSU interrupts.

3.9.4 LSU1 Interrupt Status and Clear Register

Figure 3-20. Interrupt Condition Status Register (LSU1_ICSR)

31 8	7	6	5	4	3	2	1	0
Reserved	ICS7	ICS6	ICS5	ICS4	ICS3	ICS2	ICS1	ICS0
R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-21. Interrupt Condition Clear Register (LSU1_ICCR)

31 8	7	6	5	4	3	2	1	0
Reserved	ICC7	ICC6	ICC5	ICC4	ICC3	ICC2	ICC1	ICC0
W	W	W	W	W	W	W	W	W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Where:

- ICS0 – LSU0, Transaction complete, No Errors (Posted/Non-posted),
- ICS1 – LSU1, Transaction complete, No Errors (Posted/Non-posted),
- ICS2 – LSU2, Transaction complete, No Errors (Posted/Non-posted),
- ICS3 – LSU3, Transaction complete, No Errors (Posted/Non-posted),
- ICS4 – LSU4, Transaction complete, No Errors (Posted/Non-posted),
- ICS5 – LSU5, Transaction complete, No Errors (Posted/Non-posted),
- ICS6 – LSU6, Transaction complete, No Errors (Posted/Non-posted),
- ICS7 – LSU7, Transaction complete, No Errors (Posted/Non-posted),

NOTE: Enable for these interrupts is ultimately controlled by the Interrupt Req register bit of the Load/Store command registers. This allows enabling and disabling on a per request basis. For optimum LSU performance, interrupt pacing should not be used on the LSU interrupts.

3.9.5 Error, Reset and Special Event Interrupt

Figure 3-22. Interrupt Condition Status Register (ERR_RST_EVNT_ICSR) (Address Offset 0x01E0)

31 17	16	15 12	11	10	9	8	7 3	2	1	0
Reserved	ICS16	Reserved	ICS11	ICS10	ICS9	ICS8	Reserved	ICS2	ICS1	ICS0
R	R/W	R	R/W	R/W	R/W	R/W	R	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-23. Interrupt Condition Clear Register (ERR_RST_EVNT_ICCR) (Address Offset 0x01E8)

31 17	16	15 12	11	10	9	8	7 3	2	1	0
Reserved	ICC16	Reserved	ICC11	ICC10	ICC9	ICC8	Reserved	ICC2	ICC1	ICC0
R	W	R	W	W	W	W	R	W	W	W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Where:

- ICS0 – Multicast event control symbol interrupt received on any port
- ICS1 – Port-write-In request received on any port
- ICS2 – Logical Layer Error Management Event Capture
- ICS8 – Port0 Error
- ICS9 – Port1 Error
- ICS10 – Port2 Error
- ICS11 – Port3 Error
- ICS16 – Device Reset Interrupt from any port

The interrupt status bits found in the ERR_RST_EVNT_ICSR can be cleared by writing to the ERR_RST_EVNT_ICCR register in the same manner as other interrupts. However, for new event detection and interrupt generation to occur for these special interrupts, additional register bits must be cleared.

Table 3-29 notes the additional interrupt source register bits that need to be cleared and the appropriated sequence. These are all the bits that can cause the ERR_RST_EVNT_ICSR status bits to be set.

Table 3-29. Interrupt Clearing Sequence for Special Event Interrupts

Interrupt Function	First Step	Second Step	Third Step
Multicast Event Control Symbol received on any port	Write 1 to clear: ERR_RST_EVENT[0] Offset 0x01E8	Write 1 to clear: Offset 0x1B93C EM_MECS_STAT [7:0]	Write 1 to clear: Offset 1B0D0, 0x1B150, 0x1B1D0, 0x1B250 PLM_SP(n)_RCVD_MECS [7:0]
Port Write In Request received on any port	Write 1 to clear: ERR_RST_EVENT[1] Offset 0x01E8	Write 0 to disable port writes: Offset 0x1B934 1B934 EM_DEV_PW_EN[0]	After the causes of Port-Write generation have been cleared, and the RapidIO Port {0..3} Error and Status CSR.PORT_W_P bit has been cleared if a per-port event was cleared, Write 1 to enable port writes: Offset 0x1B934 1B934 EM_DEV_PW_EN[0]
Logical Layer Error Management Capture Event	Write 1 to clear: ERR_RST_EVENT[2] Offset 0x01E8	Write all 0s to clear and unlock the capture registers: Offset 0x2008 ERR_DET	Write 0s to also possibly clear some physical layer registers. LOCAL_ERR_DET[26]- Illegal Transaction LOCAL_ERR_DET[22]- Unsupported Transaction

Table 3-29. Interrupt Clearing Sequence for Special Event Interrupts (continued)

Interrupt Function	First Step	Second Step	Third Step
Port 0 Error	Write 1 to clear: ERR_RST_EVENT[8] Offset 0x01E8	Write 1 to clear any of the following possible bits: Offset 0xB158 SP0_ERR_STAT[26] — Fatal error SP0_ERR_STAT[25] — Failed Threshold SP0_ERR_STAT[24] — Degraded Threshold Offset 0x14004 PLM_SP0_STATUS[31] — Max Retry Error	Write 1 to clear: Offset 0x1B090 PLM_SP0_STATUS[26] PLM_SP0_STATUS[25] PLM_SP0_STATUS[24] PLM_SP0_STATUS[31]
Port 1 Error	Write 1 to clear: ERR_RST_EVENT[9] Offset 0x01E8	Write 1 to clear any of the following possible bits: Offset 0xB178 SP1_ERR_STAT[26] — Fatal error SP1_ERR_STAT[25] — Failed Threshold SP1_ERR_STAT[24] — Degraded Threshold PLM_SP1_STATUS[31] — Max Retry Error	Write 1 to clear: Offset 0x1B110 PLM_SP1_STATUS[26] PLM_SP1_STATUS[25] PLM_SP1_STATUS[24] PLM_SP1_STATUS[31]
Port 2 Error	Write 1 to clear: ERR_RST_EVENT[10] Offset 0x01E8	Write 1 to clear any of the following possible bits: Offset 0xB198 SP2_ERR_STAT[26] — Fatal error SP2_ERR_STAT[25] — Failed Threshold SP2_ERR_STAT[24] — Degraded Threshold PLM_SP2_STATUS[31] — Max Retry Error	Write 1 to clear: Offset 0x1B190 PLM_SP2_STATUS[26] PLM_SP2_STATUS[25] PLM_SP2_STATUS[24] PLM_SP2_STATUS[31]
Port 3 Error	Write 1 to clear: RIO_ERR_RST_EVENT[11] Offset 0x01E8	Write 1 to clear any of the following possible bits: Offset 0xB1B8 SP3_ERR_STAT[26] — Fatal error SP3_ERR_STAT[25] — Failed Threshold SP3_ERR_STAT[24] — Degraded Threshold PLM_SP3_STATUS[31] — Max Retry Error	Write 1 to clear: Offset 0x1B210 PLM_SP3_STATUS[26] PLM_SP3_STATUS[25] PLM_SP3_STATUS[24] PLM_SP3_STATUS[31]
Device Reset	Write 1 to clear: RIO_ERR_RST_EVENT[16] Offset 0x01E8	Write 1 to clear: Offset 0x1B090, 0x1B110, 0x1B190, 0x1B210 PLM_SP(n)_STATUS [16]	Write 1 to clear: Offset 0x1B960 EM_RST_PORT_STAT [7 :0]

3.9.6 DOORBELLn Interrupt Condition Routing Registers (DOORBELLn_ICRR and DOORBELLn_ICRR2)

When doorbell packets are received by the SRIO peripheral, these ICRRs route doorbell interrupt requests from the associated doorbell ICSR to user-selected interrupt destinations. Each of the four doorbells can be mapped to these registers. The general field description in Table 3-34 applies to an ICRx field of either register.

Figure 3-24. Doorbell n Interrupt Condition Routing Register (DOORBELLn_ICCR)

31 28	27 24	23 20	19 16
ICR7	ICR6	ICR5	ICR4
RW	RW	RW	RW
15 12	11 8	7 4	3 0
ICR3	ICR2	ICR1	ICR0
RW	RW	RW	RW

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-25. Doorbell n Interrupt Condition Routing Register2 (DOORBELLn_ICCR2)

31 28	27 24	23 20	19 16
ICR15	ICR14	ICR13	ICR12
RW	RW	RW	RW
15 12	11 8	7 4	3 0
ICR11	ICR10	ICR9	ICR8
RW	RW	RW	RW

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-30. Doorbell n Interrupt Condition Routing Register Field Descriptions

Bit	Field	Description
31-0	ICRx (x = 0 to 15)	Interrupt condition routing. Routes the interrupt request from doorbell n, bit x to one of sixteen interrupt destinations (INTDST0-INTDST15). For example, If ICS6 = 1 in Doorbell 2_ICSR and ICR6 = 0010b in DOORBELL2_ICRR, the interrupt request from doorbell 2, bit 6 is sent to interrupt destination 2. • 0000b = INTDST0 • 0001b = INTDST1 • 0010b = INTDST2 • 0011b = INTDST3 • 0100b = INTDST4 • 0101b = INTDST5 • 0110b = INTDST6 • 0111b = INTDST7 • 1000b = INTDST8 • 1001b = INTDST9 • 1010b = INTDST10 • 1011b = INTDST11 • 1100b = NTDST12 • 1101b = NTDST13 • 1110b = NTDST14 • 1111b = NTDST15

3.9.7 LSU Interrupt Condition Routing Registers

Figure 3-26 shows the ICRRs for the LSU interrupt requests. The LSU0_ICSR has 32 SRCID-based interrupt sources, 16 of which are good completion interrupts and the other 16 are bad completion interrupts. Similarly, LSU1_ICSR has 8 LSU-specific interrupt sources and each interrupt is associated with an LSU. Each one of these interrupt sources have a corresponding ICRx (4 bits) configuration associated with it. This ICRx configuration specifies the interrupt routing to the different INTDSTs.

For LSU0_ICSR (32 interrupts): Interrupt routing is configured in LSU0_ICRR1-4

For LSU1_ICSR (8 interrupts): Interrupt routing is configured in LSU1_ICRR1

The complete ICRx interrupt routing to a different INTDST is shown in Table 3-31

Figure 3-26. LSU Interrupt Condition Routing Register—LSU0_ICRR1

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-27. LSU Interrupt Condition Routing Register—LSU0_ICRR2

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ICR15	ICR14	ICR13	ICR12	ICR11	ICR10	ICR9	ICR8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-28. LSU Interrupt Condition Routing Register—LSU0_ICRR3

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ICR23	ICR22	ICR21	ICR20	ICR19	ICR18	ICR17	ICR16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-29. LSU Interrupt Condition Routing Register—LSU0_ICRR4

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ICR31	ICR30	ICR29	ICR28	ICR27	ICR26	ICR25	ICR24
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-30. LSU Interrupt Condition Routing Register—LSU1_ICRR1

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
ICR7	ICR6	ICR5	ICR4	ICR3	ICR2	ICR1	ICR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-31. LSU n Interrupt Condition Routing Register Field Descriptions

Bit	Field	Description
31-0	ICRx (x = 0 to 15)	LSU condition routing. Routes the interrupt request from doorbell n, bit x to one of sixteen interrupt destinations (INTDST0-INTDST15). • 0000b = INTDST0 • 0001b = INTDST1 • 0010b = INTDST2 • 0011b = INTDST3 • 0100b = INTDST4 • 0101b = INTDST5 • 0110b = INTDST6 • 0111b = INTDST7 • 1000b = INTDST8 • 1001b = INTDST9 • 1010b = INTDST10 • 1011b = INTDST11 • 1100b = NTDST12 • 1101b = NTDST13 • 1110b = NTDST14 • 1111b = NTDST15

3.9.8 Error, Reset, and Special Event Interrupt Condition Routing Registers

The ICRRs shown in Figure 3-31 route port interrupt requests from ERR_RST_EVNT_ICSR to interrupt destinations. For example, if ICS8 = 1 in ERR_RST_EVNT_ICSR and ICR8 = 0001b in ERR_RST_EVNT_ICRR2, port 0 has generated an error interrupt request, and that request is routed to interrupt destination 1.

Figure 3-31. Error, Reset, and Special Event Interrupt Condition Routing Register—RIO_ERR_RST_EVNT_ICRR

31–12	11–8	7–4	3–0
Reserved	ICR2	ICR1	ICR0
R	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-32. Error, Reset, and Special Event Interrupt Condition Routing Register—RIO_ERR_RST_EVNT_ICRR2

31–16	15–12	11–8	7–4	3–0
Reserved	ICR11	ICR10	ICR9	ICR8
R	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Figure 3-33. Error, Reset, and Special Event Interrupt Condition Routing Register—RIO_ERR_RST_EVNT_ICRR3

31–4	3–0
Reserved	ICR16
R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

3.9.9 Interrupt Control Register

There are in total 16 interrupts for non-Doorbell type interrupts at the boundary of the peripheral. In addition to that, there are 8 more interrupts dedicated to the Doorbell interrupts only. A control bit in register INTERRUPT_CTL shown below determines whether the doorbell will be sent to the dedicated interrupts versus the general purpose 16 interrupts.

Table 3-32. INTERRUPT_CTL

Bit	Field	Reset Value	Access Type	Description
31-1	RSVD	30'b0	RO	Reserved
0	DBLL_ROUTE	0	R/W	1'b0 => RIO Doorbell Interrupts routing table is for the dedicated interrupts 1'b1 => RIO Doorbell Interrupts routing table is for the 16 general purpose interrupts

If INTERRUPT_CTL[0] is a '1', the following decoding table is used for all interrupt routing tables including the Doorbell interrupts.

Table 3-33. Interrupt Condition Routing Options for General Purpose Interrupts

Field	Access	Reset Value	Value	Function
ICRx	R	0000b	0000b	Routed to INTDST0
			0001b	Routed to INTDST1
			0010b	Routed to INTDST2
			0011b	Routed to INTDST3
			0100b	Routed to INTDST4
			0101b	Routed to INTDST5
			0110b	Routed to INTDST6
			0111b	Routed to INTDST7
			1000b	Routed to INTDST8
			1001b	Routed to INTDST9
			1010b	Routed to INTDST10
			1011b	Routed to INTDST11
			1100b	Routed to INTDST12
			1101b	Routed to INTDST13
			1110b	Routed to INTDST14
			1111b	Routed to INTDST15

If INTERRUPT_CTL[0] is a 0, Table 3-34 is used for the Doorbell interrupt routing table.

Table 3-34. Interrupt Condition Routing Options for Dedicated Doorbell Interrupts Only

Field	Access	Reset Value	Value	Function
ICRx	R	0000b	0000b	Routed to INTDST16
			0001b	Routed to INTDST17
			0010b	Routed to INTDST18
			0011b	Routed to INTDST19
			0100b	Routed to INTDST20
			0101b	Routed to INTDST21
			0110b	Routed to INTDST22
			0111b	Routed to INTDST23
			1---b	Reserved

3.9.9.1 Interrupt Status Decode Register

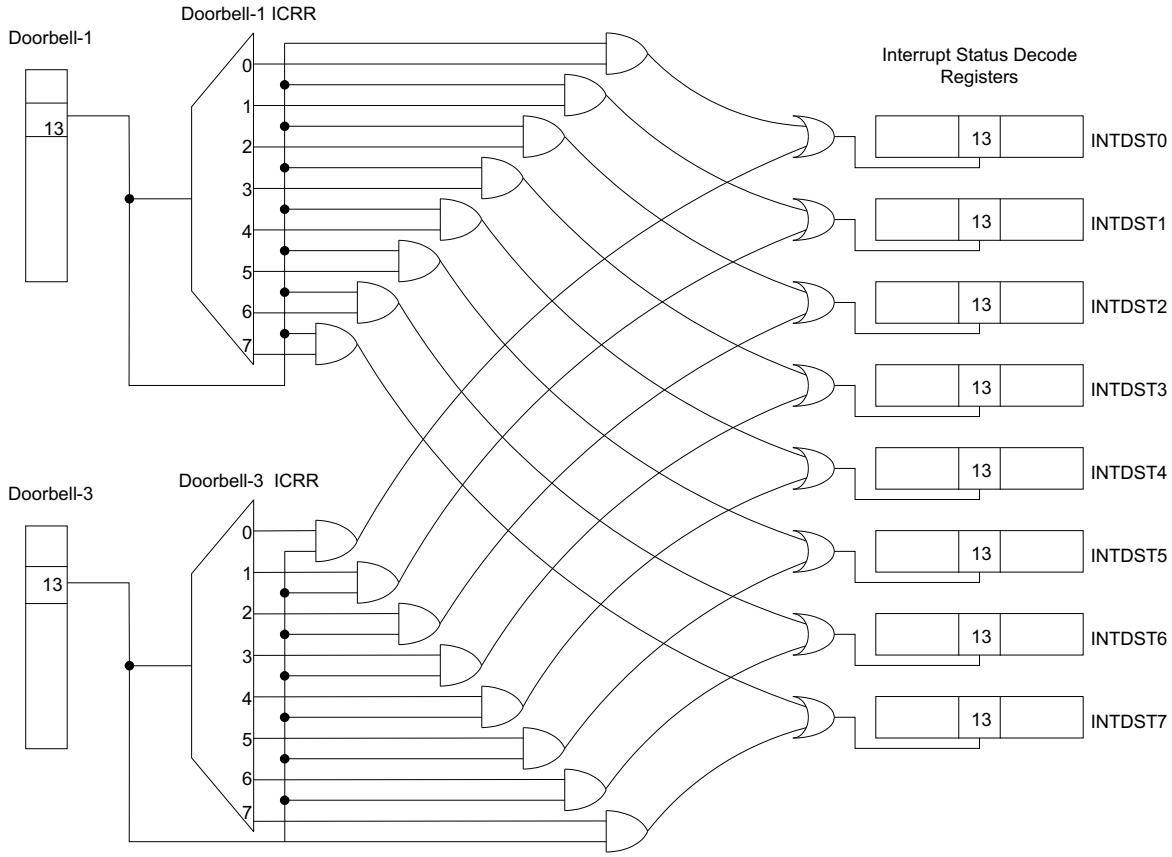
To reduce the number of reads (up to 5 reads) required finding the source bit, an Interrupt Status Decode register (ISDR) is implemented for each supported physical interrupt (INTDST0 – INTDST23). These registers are shown in Table 3-35. Interrupt sources that select a particular physical interrupt destination are mapped to specific bits in the decode register. The interrupt sources are mapped to an interrupt decode register, only if the ICRR routes the interrupt source to the corresponding physical interrupt. The status decode bit is a logical OR of multiple interrupt sources mapped to the same bit. The mapping of interrupt source bits to decode bits is fixed and non-programmable as follows:

Table 3-35. ISDRn Register

Bit	Field	Reset value	R/W	Possible interrupts routed to the bits	Description
31-21	RSVD	11'b0	RO	RSVD	Reserved
20	ERR_TYP	1'b0	RO	E	Error type interrupt has occurred, Read RIO_ERR_RST_EVNT for details
18	LSU1	1'b0	RO	LSU-1	Good completion has occurred on a LSU specific interrupt bit. This is typically used by the EDMA LSU-1 is typically used by the EDMA. the routing register routes only the relevant LSU-1 bit to the physical interrupt. As an example, if LSU3 and LSU7 are both setup for EDMA0 and EDMA1 respectively, then LSU3 interrupt is sent to a physical interrupt bit for EDMA0, while LSU7 interrupt is sent to a different physical interrupt bit, dedicated for EDMA1. The routing register ensures that the LSU interrupts are only routed to the specific physical interrupt. Thus the 2 EDMA channels will not get falsely triggered.
17-16	LSU0	2'b0	RO	LSU-0	17- SRCID specific error completion. RIO_LSU_STAT_REG0-5 must be looked at to further identify the details of the error condition 16- SRCID specific Good completion. Because the 2 bits above are tied to a specific SRCID, if the routing register routes the interrupt only to the physical interrupt dedicate to the specific SRCID, then on good completion a 2nd read will not be required.
15-0	DRBLL	16'b0	RO	Doorbell-0[15-0] Doorbell-1[15-0] Doorbell-2[15-0] Doorbell-3[15-0]	All 16 bits of each Doorbell interrupt are routed to these bits. In a typical use case, single doorbell is tied a 1 or more CPU; however, multiple doorbells will not be routed to a single CPU. Thus, a single read of the ISDR identifies precisely which doorbell bit triggered the interrupt for a specific CPU.

As an example, if both Doorbell-1 and Doorbell-3 are routed to the same physical interrupt and bit 13 of the ISDR is set, this indicates that there is a pending interrupt on either the Doorbell-1 Bit 13 or Doorbell-3 Bit 13. Figure 3-34 shows the decode routing.

Figure 3-34. Example Diagram of Interrupt Status Decode Register Mapping



LSU bits within the ICSR are logically grouped for a given LSU and OR'ed together into a single bit of the decode register. Similarly, the Error/Reset/Special event bits within the ICSR are OR'ed together into a single bit of the decode register. When either of these bits are set in the decode register, the CPU must make a additional reads to the corresponding ICSRs to determine that exact interrupt source. TI recommends that the Doorbell ICSRs are arranged per core, so that the CPU can determine the Doorbell interrupt source from a single read of the decode register.

Figure 3-35. INTDSTn_Decode Interrupt Status Decode Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ISDR31	ISDR30	ISDR29	ISDR28	ISDR27	ISDR26	ISDR25	ISDR24	ISDR23	ISDR22	ISDR21	ISDR20	ISDR19	ISDR18	ISDR17	ISDR16
R															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISDR15	ISDR14	ISDR13	ISDR12	ISDR11	ISDR10	ISDR9	ISDR8	ISDR7	ISDR6	ISDR5	ISDR4	ISDR3	ISDR2	ISDR1	ISDR0
R															

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

3.9.9.2 Interrupt Rate Counter Register

The rate at which an interrupt can be generated is controllable for each physical interrupt destination. Rate control is implemented with a programmable down-counter. The load value of the counter is written by the CPU into the registers shown in Figure 3-36. The counter re-loads and immediately starts down-counting each time the CPU writes these registers. Once the rate control counter register is written, and the counter value reaches zero (note the CPU may write zero immediately for a zero count), the interrupt pulse generation logic is allowed to fire a single pulse if any bits in the corresponding ICSR register bits are set (or become set after the zero count is reached). The counter remains at zero. Once the single pulse is generated, the logic will not generate another pulse, regardless of interrupt status changes, until the rate control counter register is written again. If interrupt pacing is not desired for a particular INTDST, it can be disabled using INTDST_RATE_DIS. If an ICSR is not mapped to an interrupt destination, pending interrupt bits within the ICSR maintain current status. Once enabled, the interrupt logic re-evaluates all pending interrupts and re-pulses the interrupt signal if any interrupt conditions are pending. The down counter is based on the DMA clock cycle.

Figure 3-36. INTDSTn_RATE_CNTL Interrupt Rate Control Counter

31	0
32-bit Count Down value	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

There are 16 registers for INTDSTn_RATE_CNTL. Interrupts 16-23 do not support this feature.

Figure 3-37. INTDST_RATE_DIS Interrupt Pacing Disable (Address Offset 0x0310)

31	30	29	28	27	26	25	24
Reserved							
R							
23	22	21	20	19	18	17	16
Reserved							
R/W							
15	14	13	12	11	10	9	8
Rate15_DIS	Rate14_DIS	Rate13_DIS	Rate12_DIS	Rate11_DIS	Rate10_DIS	Rate9_DIS	Rate8_DIS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
7	6	5	4	3	2	1	0
Rate7_DIS	Rate6_DIS	Rate5_DIS	Rate4_DIS	Rate3_DIS	Rate2_DIS	Rate1_DIS	Rate0_DIS
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

3.10 RXU Registers

The RXU registers are divided into three groups: type 11 registers, type 9 registers, and registers common to both message types. The information stored in the type 11 and type 9 registers are used to match an incoming message. The common registers hold the FlowID and destination queue ID that matching messages get assigned.

3.10.1 Type 11 Message-Mapping Registers

3.10.1.1 RIO_RXU_MAPxx_L

Figure 3-38. RIO_RXU_MAPxx_L

31 30	29 24	23 22	21 16	15 0
LTR_MASK	MBX_MASK	LTR	MBX	SRCID
R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-36. RIO_RXU_MAPxx_L

Bit	Field name	Function	Reset	Access
31-30	LTR_MASK	Mask bits for letter comparison. If a bit is • 0 => corresponding LTR bit is not used for comparision. • 1 => corresponding LTR bit is used for comparison • 00 => all letters can be mapped. • 11 => only the exact letter can be mapped	2'h3	R/W
29-24	MBX_MASK	Mask bits for mailbox comparison. If a bit is • 0 => corresponding MBX bit is not used for comparision. • 1 => corresponding MBX bit is used for comparison • 000000 => all mailboxes can be mapped. • 111111 => only the exact mailbox can be mapped	6'h3F	R/W
23-22	LTR	Letter Field. Used to compare the Letter value of the incoming packet from the UDI interface. LTR_MASK is used to further qualify the comparison.	2'h0	R/W
21-16	MBX	Mailbox Field. Used to compare the Mailbox value of the incoming packet from the UDI interface. MBX_MASK is used to further qualify the comparison. Seg_map is used to identify a single versus multisegment message. In case of multisegment message, only the bottom 2 bits of MBX are used for comparison	6'h0	R/W
15-0	SRCID	Source ID These are qualified by the tt bits described in the RIO_RXU_MAPxx_H • TT == 0 => Match only SRCID[7-0], Ignore SRCID [15-8] • TT == 1 => Match SRCID [15-0] These bits are further qualified by the SRC_PROM described in the RIO_RXU_MAPxx_H	16'h0	R/W

3.10.1.2 RIO_RXU_MAPxx_H

Figure 3-39. RIO_RXU_MAPxx_H

31 16	15	14 13	12 2	1	0
DESTID	Dest_Prom	TT	Reserved	Src_Prom	Seg_Map
R/W	R/W	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-37. RIO_RXU_MAPxx_H Field Descriptions

Bit	Field name	Function	Reset	Access
31-16	DEST_ID	Destination ID: Corresponding CPPI Destination Number	16'h0	R/W
15	DESt_PROM	Destination Promiscuous bit. • 0 -> DEST_ID must exactly match the DEST_ID in the incoming descriptor to be able access the queue • 1 -> Queue is open to any DEST_ID, no comparison is done for the DEST_ID	1'b0	R/W
14-13	TT	Transaction Type, must match the tt bits in the incoming UDI packet. • 00 -> matches the 8 LSBs of the SRCID • 01 -> matches full 16 bits of the SRCID • 10, 11 -> reserved	2'b01	R/W
12-2	Rsvd	Reserved	11'b0	RO
1	Src_Prom	Source Promiscuous bit. • 1 -> Queue is open to any SRC_ID, no comparison is done for the SRC_ID • 0 -> SRC_ID must exactly match the SRC_ID in the incoming descriptor to be able access the queue. The total bits compared are qualified by the tt bits	1'b0	R/W
0	Seg_map	Segment Mapping. If the incoming UDI packet has MSGLEN set to 1, then it indicates the incoming message is a single segment message. This information should match the Seg_map bit. • 0 — single segment (all 6 bits of the MBX are valid) • 1 — multisegment. (Only bottom 2 bits of the MBX are valid)	1'b0	R/W

3.10.2 Type 9 Message-Mapping Registers

3.10.2.1 RIO_RXU_TYPE9_MAPxx_0

Figure 3-40. RIO_RXU_TYPE9_MAPxx_0

31 24	23 16	15 0
COS_Mask	COS	SRCID
R/W	R/W	R/W

Legend: R = Read only; W = Write only

Table 3-38. RIO_RXU_TYPE9_MAPxx_0 Field Descriptions

Bit	Field name	Function	Reset	Access
31-24	COS_MASK	Mask bits for COS comparison. If a bit is • 0 => corresponding COS bit is not used for comparision. • 1 => corresponding COS bit is used for comparison • 000000 => all COS can be mapped. • 111111 => only the exact COS can be mapped	2'h0	R/W
23-16	COS	Class of Service. This field must match the incoming COS on the UDI packets. This comparison is qualified by the COS_MASK bits	8'h0	R/W
15-0	SRCID	Source ID • These are qualified by the tt bits described in the RIO_RXU_MAPxx_H • TT == 0 => Match only SRCID[7-0], Ignore SRCID [15-8] • TT == 1 => Match SRCID [15-0] • These bits are further qualified by the SRC_PROM described in the RIO_RXU_MAPxx_H	16'h0	R/W

3.10.2.2 RIO_RXU_TYPE9_MAPxx_1

Figure 3-41. RIO_RXU_TYPE9_MAPxx_1

31 16	15	14 13	12 2	1	0
DESTID	Dest_Prom	TT	Reserved	Src_Prom	Reserved
R/W	R/W	R/W	R	R/W	R

Legend: R = Read only; W = Write only

Table 3-39. RIO_RXU_TYPE9_MAPxx_1 Field Descriptions

Bit	Field name	Function	Reset	Access
31-16	DEST_ID	Destination ID: Corresponding CPPI Destination Number	16'h0	R/W
15	DESt_PROM	Destination Promiscuous bit. • 0 -> DEST_ID must exactly match the DEST_ID in the incoming descriptor to be able access the queue • 1 -> Queue is open to any DEST_ID, no comparison is done for the DEST_ID	1'b0	R/W
14-13	TT	Transaction Type, must match the tt bits in the incoming UDI packet. • 00 -> matches the 8 LSBs of the SRCID • 01 -> matches full 16 bits of the SRCID • 10, 11 -> reserved	2'b01	R/W
12-2	Rsvd	Reserved	11'b0	RO
1	Src_Prom	Source Promiscuous bit. • 1 -> Queue is open to any SRC_ID, no comparison is done for the SRC_ID • 0 -> SRC_ID must exactly match the SRC_ID in the incoming descriptor to be able access the queue. The total bits compared are qualified by the tt bits.	1'b0	R/W
0	Rsvd	Reserved	1'b0	RO

3.10.2.3 RIO_RXU_TYPE9_MAPxx_2

Figure 3-42. RIO_RXU_Type9_MAPxx_2

31 16	15 0
STRM_MASK	STRM_ID
R/W	R/W

Legend: R = Read only; W = Write only

Table 3-40. RIO_RXU_Type9_MAPxx_2 Field Descriptions

Bit	Field name	Function	Reset	Access
31-16	STRM_MASK	Mask bits for StreamID comparison. If a bit is • 0 => corresponding STRM_ID bit is not used for comparison. • 1 => corresponding STRM_ID bit is used for comparison • 00 => all Stream IDs can be mapped. • 11 => only the exact Stream ID can be mapped	16'hFFFF	R/W
15-0	STRM_ID	Stream ID These bits must match the StreamID information, which comes in the UDI packet. These bits come only in the first segment of a multisegment message.	16'h0000	R/W

3.10.3 Common Message-Mapping Registers

3.10.3.1 RIO_RXU_MAPxx_QID

Figure 3-43. RIO_RXU_MAPxx_QID

31 24	23 16	15 14	13 0
Reserved	FlowID	Reserved	Dest_QID
R	R/W	R	R/W

Legend: R = Read only; W = Write only

Table 3-41. RIO_RXU_MAPxx_QID Field Descriptions

Bit	Field name	Function	Reset	Access
31-24	Rsvd	Reserved	8'b0	RO
23-16	FlowID	Flow ID Identifies which buffer and descriptor queues should be used for storing the payload, before sending the data to the Dest_QID	8'b0	R/W
15-14	Rsvd	Reserved	2'b0	RO
13-0	Dest_QID	Destination Queue ID. This is the final destination for which the payload is targeted	14'b0	R/W

3.11 LSU and MAU Registers

The following sections describe the LSU and MAU registers.

3.11.1 LSUx_Reg0

Figure 3-44. LSUx_Reg0

31	0
RapidIO Address MSB	

Legend: R = Read only; W = Write only

Table 3-42. LSUx_Reg0 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31-0	RapidIO Address MSB	32'h0	32b *Ext Address* Fields – Packet Types 2,5, and 6

3.11.2 LSUx_Reg1

Figure 3-45. LSUx_Reg1

31	0
RapidIO Address LSB/Config_Offset	

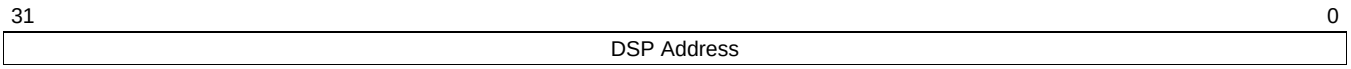
Legend: R = Read only; W = Write only

Table 3-43. LSUx_Reg1 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31-0	RapidIO Address LSB/Config_offset	32'h0	1) 32b *Address* — Packet Types 2,5, and 6 (used in conjunction with BYTE_COUNT to create 64b-aligned RapidIO packet header address) 2) 24b *Config_offset* Field — Maintenance Packets Type 8 (used in conjunction with BYTE_COUNT to create 64b-aligned RapidIO packet header Config_offset); The 2 LSB of this field must be zero because the smallest configuration access is 4B.

3.11.3 LSUx_Reg2

Figure 3-46. LSUx_Reg2



Legend: R = Read only; W = Write only

Table 3-44. LSUx_Reg2 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31-0	DSP Address	32'h0	32b DSP byte address. NA for RapidIO Header

3.11.4 LSUx_Reg3

Figure 3-47. LSU_Reg3

31	30 20	19 0
Drbll_val	Reserved	Byte_Count

Legend: R = Read only; W = Write only

Table 3-45. LSU_Reg3 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31	Drbll_val	1'b0	For FType != 10 0 -> There is no doorbell information to be sent out at the end of the packet 1 -> Doorbell information is valid If no response is required, send a doorbell after sending the last segment If response is required, after all responses are received, with no errors in them generate a doorbell with Drbll_Info If response is required, if response is received with error, doorbell will not be generated.
30-20	Reserved		
19-0	Byte_Count	19'h0	Number of data bytes to Read/Write - up to 1MB. (Used in conjunction with RapidIO address to create WRSIZE/RDSIZE and WDPTR in RapidIO packet header). • 0x0000 — 1MB • 0x0001 — 1B • 0x0010 — 2B • … • 0xFFFFF — 1048575B Maintenance requests are limited to 4B or multiple-word quantities

3.11.5 LSUx_Reg4

Figure 3-48. LSU_Reg4

31 16	15 12	11 10	9 8	7 4	3 2	1	0
DestID	SrcID_MAP	ID_Size	OutPortID	Priority	Xambs	Sup_gint	Int_Req

Legend: R = Read only; W = Write only

Table 3-46. LSU_Reg4 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31-16	DestID	16'b0	RapidIO *destinationID* field specifying target device
15-12	SrcID_MAP	4'h0	Defines which sourceID register** to use for this transaction • 0b0000 — Uses contents of RIO_DEVICEID_REG0 register • 0b0001 — Uses contents of RIO_DEVICEID_REG1 register • 0b0010 — Uses contents of RIO_DEVICEID_REG2 register • 0b0011 — Uses contents of RIO_DEVICEID_REG3 register • 0b0100 — Uses contents of RIO_DEVICEID_REG4 register • 0b0101 — Uses contents of RIO_DEVICEID_REG5 register • 0b0110 — Uses contents of RIO_DEVICEID_REG6 register • 0b0111 — Uses contents of RIO_DEVICEID_REG7 register • 0b1000 — Uses contents of RIO_DEVICEID_REG8 register • 0b1001 — Uses contents of RIO_DEVICEID_REG9 register • 0b1010 — Uses contents of RIO_DEVICEID_REG10 register • 0b1011 — Uses contents of RIO_DEVICEID_REG11 register • 0b1100 — Uses contents of RIO_DEVICEID_REG12 register • 0b1101 — Uses contents of RIO_DEVICEID_REG13 register • 0b1110 — Uses contents of RIO_DEVICEID_REG14 register • 0b1111 — Uses contents of RIO_DEVICEID_REG15 register ** Note that RIO_DeviceID_Regn are an input to the logic layer, and are not its memory map.
11-10	ID Size	2'b0	RapidIO *tt* field specifying 8- or 16-bit deviceIDs • 0b00 — 8b deviceIDs • 0b01 — 16b deviceIDs • 0b10 — reserved • 0b11 — reserved
9-8	OutPortID	2'b0	NA for RapidIO Header. Indicates the output port number for the packet to be transmitted from. Specified by the CPU and NodeID.
7-4	Priority	4'b0	Priority = {VC, PRIO[1-0], CRF} **CRF** = RapidIO CRF field used in conjunction with the Priority bit. Packets will have lower priority than a packet with same PRIO, VC bits but CRF = 1 Packets will have higher priority than a packet with same PRIO, VC bits but CRF = 0 **PRIO** = 0-3 RapidIO PRIO field specifying packet priority. Request packets should not be sent at a priority level of 3 in order to avoid system deadlock. It is the responsibility of the software to assign the appropriate out-going priority. • 00 — Priority of 0 • 01 — Priority of 1 • 10 — Priority of 2 • 11 — Priority of 3 (cannot be used by the LSU) **VC**- RapidIO virtual channel bit, VCs currently not supported 1'b0- (TI currently only supports VC0)
3-2	Xambs	2'b0	RapidIO *xambs* field specifying extended address Msb
1	SUP_GINT	1'b0	Suppress good interrupt. If interrupt request is set, and this bit is • 0— Interrupt is generated on good completion as well • 1— Interrupt is suppressed on a good completion

Table 3-46. LSU_Reg4 Field Descriptions (continued)

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
0	Interrupt Req	1'b0	NA for RapidIO Header CPU controlled request bit used for interrupt generation. Typically used in conjunction with non-posted commands to alert the CPU when the requested data and status is present. 0b0 - An interrupt is not requested upon completion of command 0b1- An interrupt is requested upon completion of command

3.11.6 LSUx_Reg5

Figure 3-49. LSU_Reg5

31 16	15 8	7 4	3 0
Drbll_Info	Hop Count	FType	TType

Legend: R = Read only; W = Write only

Table 3-47. LSU_Reg5 Field Descriptions

Bit	Control/Command Register Field	Reset value	RapidIO Packet Header Field
31-16	Drbll Info	16'h0	RapidIO doorbell info field for type 10 packets
15-8	Hop Count	8'hFF	RapidIO hop_count field specified for Type 8 maintenance packets
7-4	FType	4'b0	*ftype* field for all packets • 2 -> NREAD, Atomic instruction (TType for more details) • 5 -> NWRITE, NWRITE_R, Atomic (TType for more details) • 6 -> SWRITE • 8 -> Maintenance • 10 -> Doorbell All other encodings are reserved. If one of the reserved encoding is used, a CC of 0b100 is sent to indicate an error occured.
3-0	TType	4'b0	Transaction Field, relevant only for Ftype 2, 5, 8

3.11.7 LSUx_Reg6

RO View of the register

Figure 3-50. LSU_Reg6 (RO)

31	30	29 5	4	3 0
Busy	Full	Reserved	LCB	LTID

Legend: R = Read only; W = Write only

Table 3-48. LSU_Reg6 (RO) Field Descriptions

Bit	Field	Reset Value	Function
31	BUSY	1'b0	Indicates status of the command registers • 0b0 - Command registers are available (writable) for next set of transfer descriptors • 0b1 - Command registers are busy with current transfer
30	Full	1'b0	Indicates that all shadow registers are in use • 0b0 — At least 1 shadow register is available to be written • 0b1 — No shadow registers are available to set up any transaction
29-5	Reserved	All 0's	-
4	LCB	1'b1	LSU Context Bit. This information is used by the transaction to identify if the context of the CC is with respect to the current transaction or not.
3-0	LTID	4'b0	LSU Transaction Index. An LSU can support more than 1 transaction. This index helps identify the completion code (CC) information for the transaction.

WO View of the register

Figure 3-51. LSU_Reg6 (WO)

31 28	27	26 6	5 2	1	0
PrivID	CBUSY	Reserved	SrcID_MAP	Restart	Flush

Legend: R = Read only; W = Write only

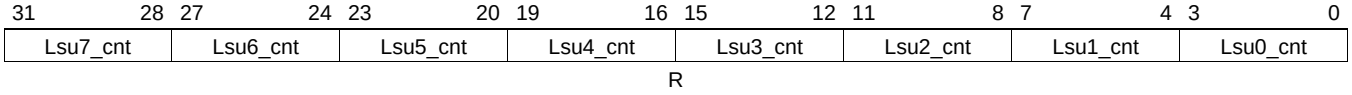
Table 3-49. LSU_Reg6 (WO) Field Descriptions

Bit	Field	Function
31-28	PrivID	When attempting to manually release the busy bit, the PrivID of the original locking CPU must match the PrivID being used for release the busy bit
27	CBusy	1 - Clear the busy bit if the PrivID matches 0 - No action
26-6	Reserved	Reserved
5-2	SrcID_MAP	Written by the software to indicate the SRCID of shadow register that need to be flushed
1	Restart	If an LSU is frozen on an error condition, a write of 1 to this bit restarts the LSU from the transaction it was working on before the error condition occurred.
0	Flush	If an LSU is frozen on an error condition, a write of 1 to this bit flushes all the shadow registers that match the SRCID in bits This bit takes higher priority over the Restart bit.

3.11.8 LSU_SETUP_REG0

Read-only view of the LSU_SETUP_REG0 register.

Figure 3-52. LSU_SETUP_REG0 (RO)



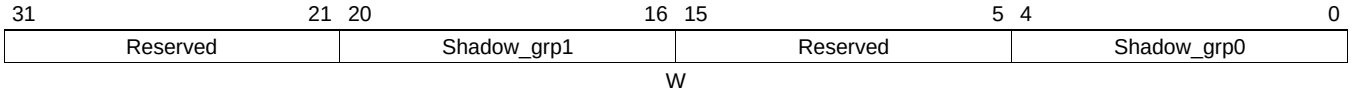
Legend: R = Read only; W = Write only

Table 3-50. LSU_SETUP_REG0 (RO) Field Descriptions

Field	Access Type	Reset Value	Description
LSUx_cnt	RO	4'h4	The total number of shadow registers associated with the LSUx This register is only programmable while the LSU logic is disabled. The read-only view reflects the total shadow registers that are associated with each. Legal range is 4'h1-4'h9. All other values are reserved.

Write-only view of the LSU_SETUP_REG0 register.

Figure 3-53. LSU_SETUP_REG0 (WO)



Legend: R = Read only; W = Write only

Table 3-51. LSU_SETUP_REG0 (WO)

Bit	Field	Access Type	Reset Value	Description
31-21	Reserved	-	0s	These bits are reserved. Any write to the reserved bits is ignored
20-16	Shadow_grp1	WO	5'h0	The total number of shadow registers associated with al the LSU 4-7 based on the table.
15-5	Reserved	-	0s	These bits are reserved. Any write to the reserved bits is ignored
4-0	Shadow_grp0	WO	5'h0	The total number of shadow registers associated with all the LSU 0-3 based on the table.

3.11.9 LSU_SETUP_REG1

Figure 3-54. LSU_SETUP_REG1

31 10	9 8	7	6	5	4	3	2	1	0
Reserved	Timeout_cnt	Lsu7_edma	Lsu6_edma	Lsu5_edma	Lsu4_edma	Lsu3_edma	Lsu2_edma	Lsu1_edma	Lsu0_edma
R					R/W				

Legend: R = Read only; W = Write only

Table 3-52. LSU_SETUP_REG1 Field Descriptions

Bit	Field	R/W	Reset	Description
31-10	Reserved	R		• Reserved
9-8	Timeout_cnt	R/W	2'h0	• 00 — After an error condition, Timecode changes only once before the LSU transaction is discarded and a new transaction is loaded from the shadow register (refer to RXU section for timecode information) • 01 — After an error condition, Timecode changes 2 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register • 10—After an error condition, Timecode changes 3 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register • 11—After an error condition, Timecode changes 4 times before the LSU transaction is discarded and a new transaction is loaded from the shadow register
7-0	LSUx_edma	R/W	1'b0	• 0—Good completion is driven to the interrupt based on the SRCID • 1—Good condition interrupt is driven to the LSU-specific interrupt bit

3.11.10 LSU_STAT_REG0

Figure 3-55. LSU_STAT_REG0

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Lsu0_Stat7	Lsu0_Stat6	Lsu0_Stat5	Lsu0_Stat4	Lsu0_Stat3	Lsu0_Stat2	Lsu0_Stat1	Lsu0_Stat0
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-53. LSU_STAT_REG0—LSUx_STATn Field Format (x = LSU ID, n = LTID)

Bit	Control/Command Register Field	Access Type	Reset value	Description
3-1	Completion Code	RO	3'b0	Indicates the status of the pending command 0b000 — Transaction complete, no errors (posted/non-posted) 0b001 — Transaction timeout occurred on non-posted transaction 0b010 — Transaction complete, packet not sent due to flow control blockade (Xoff) 0b011 — Transaction complete, non-posted response packet (type 8 and 13) contained ERROR status, or response payload length was in error 0b100 — Transaction complete, packet not sent due to unsupported transaction type or invalid programming encoding for one or more LSU register fields 0b101 — DMA data transfer error 0b110 — Retry DOORBELL response received, or Atomic test-and-swap was not allowed (semaphore in use) 0b111 — Transaction completed, no packets sent as the transaction was killed using the CBUSY bit.
0	LSU Context Bit	RO	1'b0	Gives the reference of the current CC bits. For a transaction, this must match the LCB that was returned on a read of LSU_Reg6

3.11.11 LSU_STAT_REG1

Figure 3-56. LSU_STAT_REG1

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Lsu2_Stat0	Lsu1_Stat5	Lsu1_Stat4	Lsu1_Stat3	Lsu1_Stat2	Lsu1_Stat1	Lsu1_Stat0	Lsu0_Stat8
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-53 for the register description.

3.11.12 LSU_STAT_REG2

Figure 3-57. LSU_STAT_REG2

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Lsu3_Stat3	Lsu3_Stat2	Lsu3_Stat1	Lsu3_Stat0	Lsu2_Stat4	Lsu2_Stat3	Lsu2_Stat2	Lsu2_Stat1
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-53 for the register description.

3.11.13 LSU_STAT_REG3

Figure 3-58. LSU_STAT_REG3

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Lsu4_Stat7	Lsu4_Stat6	Lsu4_Stat5	Lsu4_Stat4	Lsu4_Stat3	Lsu4_Stat2	Lsu4_Stat1	Lsu4_Stat0
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-53 for the register description.

3.11.14 LSU_STAT_REG4

Figure 3-59. LSU_STAT_REG4

31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
Lsu6_Stat0	Lsu5_Stat5	Lsu5_Stat4	Lsu5_Stat3	Lsu5_Stat2	Lsu5_Stat1	Lsu5_Stat0	Lsu4_Stat8
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-53 for the register description.

3.11.15 LSU_STAT_REG5

Figure 3-60. LSU_STAT_REG5

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
Lsu7_Stat3	Lsu7_Stat2	Lsu7_Stat1	Lsu7_Stat0	Lsu6_Stat4	Lsu6_Stat3	Lsu6_Stat2	Lsu6_Stat1
R							

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-53 for the register description.

3.11.16 LSU_FLOW_MASKS0

Figure 3-61. RIO_LSU_FLOW_MASKS0 (Address Offset 0x03EC)

31	16	15	0
LSU1 Flow Mask		LSU0 Flow Mask	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-54. LSUx Flow Mask Field Descriptions

Name	Bit	Access	Description
Flow Mask	0	R/W	0b0 — TX source doesn't support Flow0 from table entry 0b1 — TX source does support Flow0 from table entry
Flow Mask	1	R/W	0b0 — TX source doesn't support Flow1 from table entry 0b1 — TX source does support Flow1 from table entry
Flow Mask	2	R/W	0b0 — TX source doesn't support Flow2 from table entry 0b1 — TX source does support Flow2 from table entry
Flow Mask	3	R/W	0b0 — TX source doesn't support Flow3 from table entry 0b1 — TX source does support Flow3 from table entry
Flow Mask	4	R/W	0b0 — TX source doesn't support Flow4 from table entry 0b1 — TX source does support Flow4 from table entry
Flow Mask	5	R/W	0b0 — TX source doesn't support Flow5 from table entry 0b1 — TX source does support Flow5 from table entry
Flow Mask	6	R/W	0b0 — TX source doesn't support Flow6 from table entry 0b1 — TX source does support Flow6 from table entry
Flow Mask	7	R/W	0b0 — TX source doesn't support Flow7 from table entry 0b1 — TX source does support Flow7 from table entry
Flow Mask	8	R/W	0b0 — TX source doesn't support Flow8 from table entry 0b1 — TX source does support Flow8 from table entry
Flow Mask	9	R/W	0b0 — TX source doesn't support Flow9 from table entry 0b1 — TX source does support Flow9 from table entry
Flow Mask	10	R/W	0b0 — TX source doesn't support Flow10 from table entry 0b1 — TX source does support Flow10 from table entry
Flow Mask	11	R/W	0b0 — TX source doesn't support Flow11 from table entry 0b1 — TX source does support Flow11 from table entry
Flow Mask	12	R/W	0b0 — TX source doesn't support Flow12 from table entry 0b1 — TX source does support Flow12 from table entry
Flow Mask	13	R/W	0b0 — TX source doesn't support Flow13 from table entry 0b1 — TX source does support Flow13 from table entry
Flow Mask	14	R/W	0b0 — TX source doesn't support Flow14 from table entry 0b1 — TX source does support Flow14 from table entry
Flow Mask	15	R/W	0b0 — TX source doesn't support Flow15 from table entry 0b1 — TX source does support Flow15 from table entry

3.11.17 LSU_FLOW_MASKS1

Figure 3-62. RIO_LSU_FLOW_MASKS1 (Address Offset 0x03F0)

31 16	15 0
LSU3 Flow Mask	LSU2 Flow Mask
R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-54 for the register description.

3.11.18 LSU_FLOW_MASKS2

Figure 3-63. RIO_LSU_FLOW_MASKS2 (Address Offset 0x03F4)

31 16	15 0
LSU5 Flow Mask	LSU4 Flow Mask
R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-54 for the register description.

3.11.19 LSU_FLOW_MASKS3

Figure 3-64. RIO_LSU_FLOW_MASKS3 (Address Offset 0x03F8)

31	16	15	0
LSU7 Flow Mask		LSU6 Flow Mask	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

See Table 3-54 for the register description.

3.11.20 SUPRVSR_ID (MAU)

Figure 3-65. SUPRVSR_ID (MAU)

31 24	23 16	15 0
Reserved	8b SUPRVSR_ID	16b SUPRVSR_ID
R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-55. SUPRVSR_ID (MAU) Field Descriptions

Bit	Name	Access	Reset Source	Reset Value	Description
31-24	Reserved	R		0x0	
23-16	8b SUPRVSR_ID	R/W	VBUS_rst	0x00	Supervisor permissions ID compared against incoming packet's SOURCEIDs
15-0	16b SUPRVSR_ID	R/W	VBUS_rst	0x0000	Supervisor permissions ID compared against incoming packet's SOURCEIDs

3.12 Flow Control Registers

The following sections describe the flow control registers.

3.12.1 FLOW_CNTLx

Figure 3-66. FLOW_CNTLx

31 18	17 16	15 0
Reserved	tt	Flow_Cntrl_IDx
R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-56. FLOW_CNTLx Field Descriptions

Bit	Name	Access	Reset Source	Reset Value	Description
31-18	Reserved	R			Reserved
17-16	tt	R/W	VBUS_rst	0b01	Selects Flow_Cntl_ID length. • 0b00 – 8b IDs • 0b01 – 16b IDs • 0b10 – 0b11 — reserved
15-0	Flow_Cntl_ID0	R/W	VBUS_rst	0x0000	DestID of Flow 0
15-0	Flow_Cntl_ID1	R/W	VBUS_rst	0x0000	DestID of Flow 1
15-0	Flow_Cntl_ID2	R/W	VBUS_rst	0x0000	DestID of Flow 2
15-0	Flow_Cntl_ID3	R/W	VBUS_rst	0x0000	DestID of Flow 3
15-0	Flow_Cntl_ID4	R/W	VBUS_rst	0x0000	DestID of Flow 4
15-0	Flow_Cntl_ID5	R/W	VBUS_rst	0x0000	DestID of Flow 5
15-0	Flow_Cntl_ID6	R/W	VBUS_rst	0x0000	DestID of Flow 6
15-0	Flow_Cntl_ID7	R/W	VBUS_rst	0x0000	DestID of Flow 7
15-0	Flow_Cntl_ID8	R/W	VBUS_rst	0x0000	DestID of Flow 8
15-0	Flow_Cntl_ID9	R/W	VBUS_rst	0x0000	DestID of Flow 9
15-0	Flow_Cntl_ID10	R/W	VBUS_rst	0x0000	DestID of Flow 10
15-0	Flow_Cntl_ID11	R/W	VBUS_rst	0x0000	DestID of Flow 11
15-0	Flow_Cntl_ID12	R/W	VBUS_rst	0x0000	DestID of Flow 12
15-0	Flow_Cntl_ID13	R/W	VBUS_rst	0x0000	DestID of Flow 13
15-0	Flow_Cntl_ID14	R/W	VBUS_rst	0x0000	DestID of Flow 14
15-0	Flow_Cntl_ID15	R/W	VBUS_rst	0x0000	DestID of Flow 15, if all 0s this table entry represents all flows other than flows 0 - 14

3.13 TXU Registers

The following sections describe the TXU registers.

3.13.1 TX_CPPI_FLOW_MASKSx

Table 3-57. TX_CPPI_FLOW_MASKSx

Name of Register	Address Offset	Reset Value	Bits[31-16]	Bits[15-0]
RIO_TX_CPPI_FLOW_MASKS0	0x0420	32'hFFFFFFFF	TX Queue1 Flow Mask	TX Queue0 Flow Mask
RIO_TX_CPPI_FLOW_MASKS1	0x0424	32'hFFFFFFFF	TX Queue3 Flow Mask	TX Queue2 Flow Mask
RIO_TX_CPPI_FLOW_MASKS2	0x0428	32'hFFFFFFFF	TX Queue5 Flow Mask	TX Queue4 Flow Mask
RIO_TX_CPPI_FLOW_MASKS3	0x042C	32'hFFFFFFFF	TX Queue7 Flow Mask	TX Queue6 Flow Mask
RIO_TX_CPPI_FLOW_MASKS4	0x0430	32'hFFFFFFFF	TX Queue9 Flow Mask	TX Queue8 Flow Mask
RIO_TX_CPPI_FLOW_MASKS5	0x0434	32'hFFFFFFFF	TX Queue11 Flow Mask	TX Queue10 Flow Mask
RIO_TX_CPPI_FLOW_MASKS6	0x0438	32'hFFFFFFFF	TX Queue13 Flow Mask	TX Queue12 Flow Mask
RIO_TX_CPPI_FLOW_MASKS7	0x043C	32'hFFFFFFFF	TX Queue15 Flow Mask	TX Queue14 Flow Mask

Table 3-58. Flow Mask Field Descriptions

Bit	Name	Access	Description
0	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow0 from table entry • 0b1 — TX source does support Flow0 from table entry
1	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow1 from table entry • 0b1 — TX source does support Flow1 from table entry
2	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow2 from table entry • 0b1 — TX source does support Flow2 from table entry
3	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow3 from table entry • 0b1 — TX source does support Flow3 from table entry
4	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow4 from table entry • 0b1 — TX source does support Flow4 from table entry
5	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow5 from table entry • 0b1 — TX source does support Flow5 from table entry
6	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow6 from table entry • 0b1 — TX source does support Flow6 from table entry
7	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow7 from table entry • 0b1 — TX source does support Flow7 from table entry
8	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow8 from table entry • 0b1 — TX source does support Flow8 from table entry
9	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow9 from table entry • 0b1 — TX source does support Flow9 from table entry
10	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow10 from table entry • 0b1 — TX source does support Flow10 from table entry
11	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow11 from table entry • 0b1 — TX source does support Flow11 from table entry
12	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow12 from table entry • 0b1 — TX source does support Flow12 from table entry
13	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow13 from table entry • 0b1 — TX source does support Flow13 from table entry

Table 3-58. Flow Mask Field Descriptions (continued)

Bit	Name	Access	Description
14	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow14 from table entry • 0b1 — TX source does support Flow14 from table entry
15	Flow Mask	R/W	• 0b0 — TX source doesn't support Flow15 from table entry • 0b1 — TX source does support Flow15 from table entry

3.13.2 TX_QUEUE_SCH_INFOx

Table 3-59. TX_QUEUE_SCH_INFOx

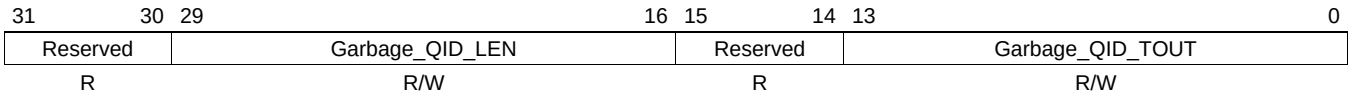
Register	31-24	23-16	15-8	7-0
RIO_TX_QUEUE_SCH_INFO1	Queue3_info	Queue2_info	Queue1_info	Queue0_info
RIO_TX_QUEUE_SCH_INFO2	Queue7_info	Queue6_info	Queue5_info	Queue4_info
RIO_TX_QUEUE_SCH_INFO3	Queue11_info	Queue10_info	Queue9_info	Queue8_info
RIO_TX_QUEUE_SCH_INFO4	Queue15_info	Queue14_info	Queue13_info	Queue12_info

Table 3-60. QueueN_info Field Descriptions

Bit	Field	Implies	Reset Value	Access Type	Description
7-6	Reserved	Reserved	RO	2'b0	Reserved
5-4	QueueN_Port	Port Map	2'b0	R/W	The port to which the data will be transmitted to • 00 — Data to Port0 • 01 — Data to Port1 • 10 — Data to Port2 • 11 — Data to Port3
3	Priority [3]	VC Bit	1'b0	R/W	Not currently supported
2-1	Priority [2-1]	00 => PRIO 0 01 => PRIO 1 10 => PRIO 2 11 => PRIO 3	2'b00	RO	PRIO 0 is lowest, and 3 the highest PRIO 3 cannot be used for TXU as the message response PRIO has to be 1 higher than the transmit message to prevent deadlock The priority information comes from the PKTDMA
0	Priority [0]	CRF Bit	1'b0	R/W	CRF by itself does not have any buffering resources dedicated to it. If this bit is set, this queue will get scheduled before another queue at same PRIO and CRF=0

3.13.3 Garbage_Coll_QID0

Figure 3-67. RIO_Garbage_Coll_QID0



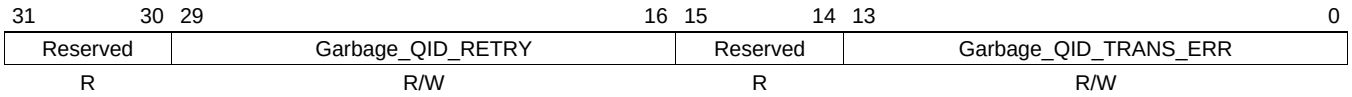
Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-61. RIO_Garbage_Coll_QID0 Field Descriptions

Bit	Field name	Function	Reset	Access
31-30	Reserved	Reserved	2'b0	R
29-16	Garbage _QID_LEN	Length mismatch between size in the UDI packet and received payload	18'h0	R/W
15-14	Reserved	Reserved	2'b0	R
13-0	Garbage _QID_TOUT	TimeOut on receiving one of the segments	14'h0	R/W

3.13.4 Garbage_Coll_QID1

Figure 3-68. Garbage_Coll_QID1



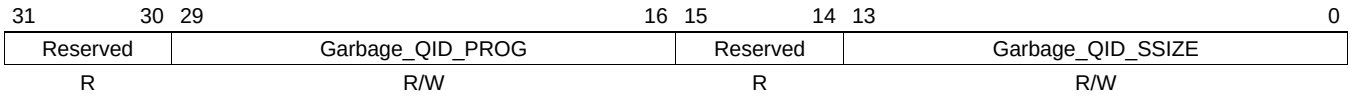
Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-62. Garbage_Coll_QID1 Field Descriptions

Bit	Field name	Function	Reset	Access
31-30	Reserved	Reserved	2'b0	R
29-16	GARBAGE_QID_RETRY	Excessive Retries. Message received more than retry_count retry responses.	14'h0	R/W
15-14	RSVD	Reserved	2'b0	R
13-0	GARBAGE_QID_TRANS_ERR	Transaction error. Message received an error response. An ERROR transfer completion code indicates an error in one or more segments of a transmitted multisegment message. These are written by the SRIO when releasing the descriptor and not the master that initiated the transfer	14'h0	R/W

3.13.5 Garbage_Coll_QID2

Figure 3-69. Garbage_Coll_QID2



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-63. Garbage_Coll_QID2 Field Descriptions

Bit	Field name	Function	Reset	Access
31-30	RSVD	Reserved	2'b0	R
29-16	Garbage_QID_PROG	Garbage queue for identifying all programming errors.	14'h0	R/W
15-14	RSVD	Reserved	2'b0	R
13-0	GARBAGE_QID_SSIZE	In type11, Message_length/16 must be less than or equal to Ssize. If this requirement is not met, the message is not sent. Instead the descriptor is sent to this QID.	14'h0	R/W

3.14 PKTDMA Registers

PKTDMA Global Control Registers, PKTDMA Tx DMA Channel Control/Status Registers, PKTDMA Rx DMA Channel Control/Status Registers, PKTDMA Tx DMA Scheduler Configuration Registers and PKTDMA Rx DMA Flow Configuration Registers are part of the *Multicore Navigator User's Guide*.

See the *Multicore Navigator User's Guide* in Related Documentation from Texas Instruments for detailed information about the bit fields in these registers.

3.15 CSR and CAR Registers

The following sections describe the SRIO CSR/CAR registers.

3.15.1 DEV_ID

The device identity CAR (DEV_ID) is shown in Figure 3-70 and described in Table 3-64. Writes have no effect to this register after boot_complete is set. This register is programmed by the Boot ROM if SRIO boot mode is used. Otherwise, they must be programmed by the init function. DeviceIdentity field value should be programmed to be 0x009D.

Figure 3-70. Device Identity CAR

31	16	15	0
DeviceIdentity		Device VendorIdentity	
R		R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-64. Device Identity CAR Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description	Comments
31-16	DeviceIdentity	R	VBUS_rst	0x0000	Identifies the type of device. Vendor specific	These 16b should match the 16b from the JTAG PartID field of the DeviceID register This is programmed through the SRIO boot configuration from the ROM
15-0	Device VendorIdentity	R	VBUS_rst	0x0030	Device Vendor ID	Assigned by RapidIO TA

3.15.2 DEV_INFO

The device information CAR (DEV_INFO) is shown in Figure 3-71. This register is programmed by the Boot ROM if SRIO boot mode is used. Otherwise, they must be programmed by the init function.

Figure 3-71. Device Information CAR

31	0
Devisers	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-65. Device Information CAR Field Descriptions

Bit	Name	R/W	Reset Value	Description	Described In Section
31-0	Devisers	R/W	32'h0	Vendor supply device revision	The lower 4b should match the 4b from the JTAG Variant field of the DeviceID register. This is programmed through the SRIO boot configuration from the ROM.

3.15.3 ASBLY_ID

The assembly identity CAR (ASBLY_ID) is shown in Figure 3-72 and described in Table 3-66. Writes have no effect to this register. The values are hard coded and will not change from their reset state.

Figure 3-72. Assembly Identity CAR—ASBLY_ID

31 16	15 0
ASSY_IDENTITY	ASSY_VENDORIDENTITY
R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

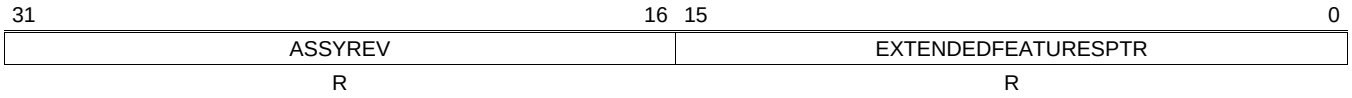
Table 3-66. Assembly Identity CAR—ASBLY_ID Field Descriptions

Bit	Field	Value	Description
31-16	ASSY_IDENTITY	0000h	Assembly identifier. Vendor-specific.
15-0	ASSY_VENDORIDENTITY	0030h	Assembly vendor identifier assigned by RapidIO Trade Association.

3.15.4 ASBLY_INFO

The assembly Information CAR (ASBLY_INFO) is shown in Figure 3-73 and described in Table 3-67. Writes have no effect to this register. The values are hard coded and do not change from their reset state.

Figure 3-73. Assembly Information CAR—ASBLY_INFO



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

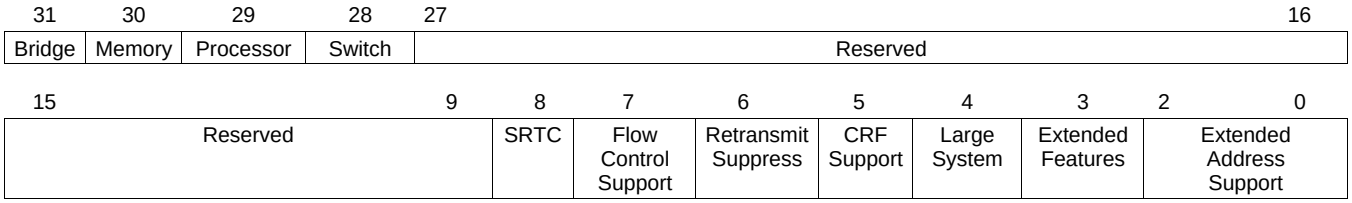
Table 3-67. Assembly Information CAR—ASBLY_INFO Field Descriptions

Bit	Field	Value	Description
31-16	ASSYREV	0000h	Assembly revision level.
15-0	EXTENDEDFEATURESPTR	0100h	Pointer to first entry in extended features list.

3.15.5 PE_FEAT

The processing element features CAR (PE_FEAT) is shown in Figure 3-74 and described in Table 3-68.

Figure 3-74. Processing Element Features CAR (PE_FEAT)



LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 3-68. Processing Element Features CAR (PE_FEAT) Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description
31	Bridge	R	VBUS_rst	0b0	PE can bridge to another interface. Examples are PCI, proprietary processor buses, DRAM, and so forth. 0b0 - Device in not a bridge 0b1 - Device can bridge to another interface
30	Memory	R	VBUS_rst	0b0	PE has physically addressable local address space and can be accessed as an end point through non-maintenance (such as non-coherent read and write) operations. This local address space may be limited to local configuration registers, or could be on-chip SRAM. 0b0 - Device is not a RapidIO-endpoint addressable for reads and writes 0b1 - Device has physically addressable local address space and can be accessed as an endpoint through logical I/O (that is, NREAD and NWRITE) transactions.
29	Processor	R	VBUS_rst	0b1	PE physically contains a local processor or similar device that executes code. A device that bridges to an interface that connects to a processor does not count (see bit 31 above). 0b0 - Device is not a processor 0b1 - Device physically contains a local processor or similar device that executes code. A device that bridges to an interface that connects to a processor does not count (see bit 0 above).
28	Switch	R	VBUS_rst	0b0	PE can bridge to another external RapidIO interface - an internal port to a local end point does not count as a switch port. For example, a device with two RapidIO ports and a local end point is a two port switch, not a three port switch, regardless of the internal architecture. 0b0 - Device is not a switch 0b1 - Device is a switch. FType 8 packets with hop count equal to 0 are routed to the register bus.
27-9	Reserved			0b0	Reserved
8	SRTC	R	VBUS_rst	0b1	• Standard routing table configuration support • 0b0 - Device does not support the standard route table configuration • method • 0b1 - Device supports the standard route table configuration method
7	Flow Control Support	R	VBUS_rst	0b0	PE supports congestion flow control mechanism 0b0 - Device does not support the flow control extension 0b1 - Device does support the flow control extension
6	Re-transmit Suppression Support	R	VBUS_rst	0b0	PE supports suppression of error recovery on packet CRC errors. 0b1 - PE supports suppression
5	CRF Support	R	VBUS_rst	0b0	This bit indicates PE support for the Critical Request Flow (CRF) function 0b1 - PE supports CRF function

Table 3-68. Processing Element Features CAR (PE_FEAT) Field Descriptions (continued)

Bit	Name	R/W	Reset Source	Reset Value	Description
4	Common Transport Large System Support	R	VBUS_rst	0b0	The srcID and destID of endpoint are generated in accordance with this bit. 0b0 - Device supports 8-bit destIDs only 0b1 - Device supports 8- and 16-bit destIDs
3	Extended Features	R	VBUS_rst	0b1	PE has extended features list; the extended features pointer is valid
2-0	Extended addressing support	R	VBUS_rst	0b001	Indicates the number address bits supported by the PE both as a source and target of an operation. All PEs shall at minimum support 34 bit addresses. 0b111 - PE supports 66-, 50-, and 34-bit addresses 0b101 - PE supports 66- and 34-bit addresses 0b011 - PE supports 50- and 34-bit addresses 0b001 - PE supports 34-bit addresses

3.15.6 SW_PORT

The switch port information CAR (SW_PORT) is shown in Figure 3-75 and described in Table 3-69.

Figure 3-75. Switch Port Information CAR (SW_PORT)

31 16	15 8	7 0
Reserved	Port Total	Port Num
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-69. Switch Port Information CAR (SW_PORT) Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	Reserved	R		0b0	PE can bridge to another interface. Examples are PCI, proprietary processor buses, DRAM, and so forth.
15-8	Port Total	R	VBUS_rst	0h04	The total number of RapidIO ports on the device. 0x00 = Reserved 0x01 = 1 port ... 0xFF = 255 ports The total number of RapidIO ports on the device. . If in 4x/2x mode the device has less ports than in 1x mode, the PORT_TOTAL has a number of ports defined for 1x mode.
7-0	Port Num	R	VBUS_rst	0b0	The port number from which the maintenance read operation accessed this register. When read from a non-RapidIO register bus master, the value returned is 0.

3.15.7 SRC_OP

The source operations CAR (SRC_OP) is shown in Figure 3-76 and described in Table 3-70.

Figure 3-76. Source Operations CAR—SRC_OP

31 20	19	18	17 16	15 14	13	12	11	10
Rsvd	DS_TM	DS	Implmnt Defined	Read	Write	Stream Write	Write W/Resp	Data Mess	Door Bell
				R				

9	8	7	6	5	4	3	2	1 0
Reserved	Atomic (tst/swap)	Atomic (incrmnt)	Atomic (dcrmnt)	Atomic (set)	Atomic (clear)	Reserved	Port-Write	Implmnt Defined
				R				

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-70. Source Operations CAR Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description
31-20	Reserved	R	VBUS_rst	0b0	Reserved
19	DS_TM	R	VBUS_rst	0b0	Supports data streaming traffic management
18	DS	R	VBUS-rst	0b1	Supports data streaming
17-16	Implementation Defined	R	VBUS_rst	0b0	Defined by the device implementation
15	Read	R	VBUS_rst	0b0	PE can support a read operation
14	Write	R	VBUS_rst	0b0	PE can support a write operation
13	Streaming-write	R	VBUS_rst	0b0	PE can support a streaming-write operation
12	Write-with-response	R	VBUS_rst	0b0	PE can support a write-with-response operation
11	Data Message	R	VBUS_rst	0b0	PE can support a data message operation
10	DoorBell	R	VBUS_rst	0b0	PE can support a doorbell operation
9	Reserved				Reserved
8	Atomic (test-and-swap)	R	VBUS_rst	0b0	PE can support an atomic test-and-swap operation
7	Atomic (increment)	R	VBUS_rst	0b0	PE can support an atomic increment operation
6	Atomic (decrement)	R	VBUS_rst	0b0	PE can support an atomic decrement operation
5	Atomic (set)	R	VBUS_rst	0b0	PE can support an atomic set operation
4	Atomic (clear)	R	VBUS_rst	0b0	PE can support an atomic clear operation
3	Reserved	R	VBUS_rst	0b0	Reserved
2	Port-write	R	VBUS_rst	0b1	PE can support a port-write generation
1-0	Implementation Defined	R	VBUS_rst	0b00	Defined by the device implementation

3.15.8 DEST_OP

The destination operations CAR (DEST_OP) is shown in Figure 3-77 and described in Table 3-71.

Figure 3-77. Destination Operation CAR—DEST_OP

31–18	17–16	15	14	13	12	11	10
Reserved	Implmnt Defined	Read	Write	Stream Write	Write W/Resp	Data Mess	Door Bell
		R					

9	8	7	6	5	4	3	2	1–0
Rsvd	Atomic (tst/swap)	Atomic (incrmnt)	Atomic (dcrmnt)	Atomic (set)	Atomic (clear)	Reserved	Port-Write	Implmnt Defined
				R				

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-71. Destination Operation CAR Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description
31-18	Reserved	R	VBUS_rst	0b0	Reserved
17-16	Implementation Defined	R	VBUS_rst	0b0	Defined by the device implementation
15	Read	R	VBUS_rst	0b0	PE can support a read operation
14	Write	R	VBUS_rst	0b0	PE can support a write operation
13	Streaming-write	R	VBUS_rst	0b0	PE can support a streaming-write operation
12	Write-with-response	R	VBUS_rst	0b0	PE can support a write-with-response operation
11	Data Message	R	VBUS_rst	0b0	PE can support a data message operation
10	DoorBell	R	VBUS_rst	0b0	PE can support a doorbell operation
9	Reserved				Reserved
8	Atomic (test-and-swap)	R	VBUS_rst	0b0	PE can support an atomic test-and-swap operation
7	Atomic (increment)	R	VBUS_rst	0b0	PE can support an atomic increment operation
6	Atomic (decrement)	R	VBUS_rst	0b0	PE can support an atomic decrement operation
5	Atomic (set)	R	VBUS_rst	0b0	PE can support an atomic set operation
4	Atomic (clear)	R	VBUS_rst	0b0	PE can support an atomic clear operation
3	Reserved	R	VBUS_rst	0b0	Reserved
2	Port-write	R	VBUS_rst	0b1	PE can support a port-write operation
1-0	Implementation Defined	R	VBUS_rst	0b00	Defined by the device implementation

3.15.9 DS_INFO

This register defines the data-streaming capabilities of a processing element.

Figure 3-78. DS_INFO

31 16	15 0
MaxPDU	SegSupport

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-72. DS_INFO Field Descriptions

Bit	Field	Reset Value	Reset By	Description
31-16	MaxPDU	16'b0	Vbusp_rstn	Maximum PDU- The maximum PDU size in bytes supported by the destination end point 0x0000 — 64K bytes 0x0001 — 1 byte 0x0002 — 2 bytes . . 0xFFF — 64Kbytes-1
15-0	SegSupport	16'h0010	Vbusp_rstn	Segmentation Support- The number of segmentation contexts supported by the destination endpoint 0x0000 — 64K segmentation contexts 0x0001 — 1 segmentation context 0x0002 — 2 segmentation context . . 0xFFF — 64K-1 segmentation context

3.15.10 DS_LL_CTL

The data streaming logical layer control CSR is used for general command and status information for the logical interface.

Figure 3-79. DS_LL_CTL

31 28	27 24	23 8	7 0
TM Types Supported	TM Mode	Reserved	MTU
R	R/W		

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-73. DS_LL_CTL Field Descriptions

Bit	Field	R/W	Reset value	Description
31-28	TM Types Supported	RO	4'b0	Bit 0 — 1, Basic type supported Bit 1 — 1, Rate type supported Bit 2 — 1, Credit type supported Bit 3— reserved Valid combination 0b0001, 0b0011, 0b0101, 0b0111. All others are invalid Because TM mode is disabled, this is not valid
27-24	TM Mode	R/W	4'b0	Traffic Management mode of operation 0b0000 = TM disabled 0b0001 = Basic 0b0010 = Rate 0b0011 = Credit 0b0100 = Credit + rate 0b0101 - 0b0111 = Reserved 0b1000 - 0b1111 = Allowed for user-defined modes TI only supports disabled
23-8	Reserved			
7-0	MTU		0b0100_0000	Maximum Transmission Unit - controls the data payload size for segments of an encapsulated PDU. Only single-segment PDUs and end segments are permitted to have a data payload that is less than this value. The MTU can be specified in increments of 16 bytes. 0b0000_0100 — 16-byte block size 0b0000_1000 — 32-byte block size 0b0000_1100 — 48-byte block size 0b0001_0000 — 64-byte block size 0b0001_0100 — 80-byte block size 0b0001_1000 — 96-byte block size 0b0001_1100 — 112-byte block size 0b0010_0000 — 128-byte block size ... 0b0100_0000 — 256-byte block size All other encodings reserved

3.15.11 PE_LL_CTL

The processing element logical layer control CSR (PE_LL_CTL) is shown in Figure 3-80 and described in Table 3-74.

Figure 3-80. Processing Element Logical Layer Control CSR]—PE_LL_CTL

31	3	2	0
RESERVED		EXTENDED ADDRESSING CONTROL	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-74. Processing Element Logical Layer Control CSR]—PE_LL_CTL Field Descriptions

Bit	Field	Description
31-3	Reserved	These read-only bits return 0s when read.
2-0	EXTENDED_ADDRESSING_CONTROL	Controls the number of address bits generated by the PE as a source and processed by the PE as the target of an operation. All other encodings are reserved. • 001b = PE supports 34-bit addresses • 010b = PE supports 50-bit addresses • 100b = PE supports 66-bit addresses • Other = Reserved

3.15.12 LCL_CFG_HBAR

The local configuration space base address 0 CSR (LCL_CFG_HBAR) is shown in Figure 3-81 and described in Table 3-75.

Figure 3-81. Local Configuration Space Base Address 0 CSR—LCL_CFG_HBAR

31	30	0
Reserved	LCSBA	
R	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-75. Local Configuration Space Base Address 0 CSR—LCL_CFG_HBAR Field Descriptions

Bit	Field	Value	Description
31	Reserved	0	These read-only bits return 0s when read.
30-0	LCSBA	00000000h to FFFFFFFFh	Bits 30 to 15 are reserved for 34-bit addresses, reserved for 50-bit addresses, and bits 66 to 51 of a 66-bit address. Bits 14 to 0 are reserved for 34-bit addresses, bits 50 to 36 of a 50-bit address, and bits 50 to 36 of a 66-bit address.

3.15.13 LCL_CFG_BAR

The local configuration space base address 1 CSR (LCL_CFG_BAR) is shown in Figure 3-82 and described in Table 3-76.

Figure 3-82. Local Configuration Space Base Address 1CSR—LCL_CFG_BAR

31	0
LCSBA	
R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-76. Local Configuration Space Base Address 1CSR—LCL_CFG_BAR Field Descriptions

Bit	Field	Value	Description
31-0	LCSBA	00000000h to FFFFFFFFh	Bit 31 is reserved for 34-bit addresses, bit 35 of a 50-bit address, and bits 35 of a 66-bit address. Bits 30 to 0 are bits 34 to 3 of a 34-bit address, bits 35 to 3 of a 50-bit address, and bits 35 to 3 of a 66-bit address.

3.15.14 BASE_ID

The base device ID CSR (BASE_ID) is shown in Figure 3-83 and described in Table 3-77.

Figure 3-83. Base Device ID CSR—BASE_ID

31 24	23 16	15 0
Reserved	BASE_DEVICEID	LARGE_BASE_DEVICEID
R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-77. Base Device ID CSR—BASE_ID Field Descriptions

Bit	Field	Value	Description
31-24	Reserved	00h	These read-only bits return 0s when read.
23-16	BASE_DEVICEID	FFh	This is the base ID of the device in small common transport system (endpoints only).
15-0	ASSY_VENDORIDENTITY	0000h-FFFFh	This is the base ID of the device in large common transport system (only valid for endpoints and if bit 4 of the PE_FEAT Register is set).

3.15.15 HOST_BASE_ID_LOCK

See Section 2.4.2 of the RapidIO *Common Transport Specification* for a description of this register. It provides a lock function that is write-once/resetable. The host base device ID lock CSR (HOST_BASE_ID_LOCK) is shown in Figure 3-84 and described in Table 3-78.

Figure 3-84. Host Base Device ID Lock CSR

31	16	15	0
Reserved		HOST_BASE_DEVICEID	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

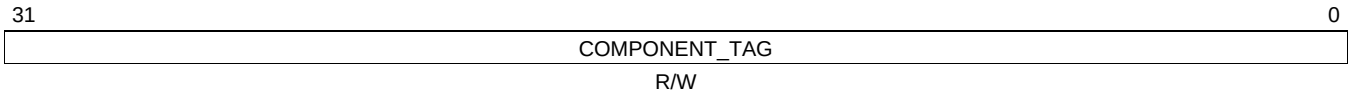
Table 3-78. Host Base Device ID Lock CSR Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0000h	These read-only bits return 0s when read.
15-0	HOST_BASE_DEVICEID	0000h-ffffh	This is the base ID for the Host PE that is initializing this PE.

3.15.16 COMP_TAG

The component Tag CSR (COMP_TAG) is shown in Figure 3-85 and described in Table 3-79.

Figure 3-85. Component Tag CSR—COMP_TAG



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-79. Component Tag CSR—COMP_TAG Field Descriptions

Bit	Field	Value	Description
31-0	COMPONENT_TAG	00000000h to FFFFFFFFh	Software-defined component tag for the PE. Useful for devices without device IDs.

3.15.17 SP_MB_HEAD

The 1x/4x LP_Serial port maintenance block header register (SP_MB_HEAD) is shown in Figure 3-86 and described in Table 3-80.

Figure 3-86. Port Maintenance Block Header Register—SP_MB_HEAD

31	16	15	0
EF_PTR		EF_ID	
R		R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-80. 1x/4x LP_Serial Port Maintenance Block Header Register (SP_MD_HEAD) Field Description

Bit	Field	Description
31-16	EF_PTR	Hard-wired pointer to the next block in the data structure.
15-0	EF_ID	Hard-wired extended features ID. • 0001h = General endpoint device • 0002h = General endpoint device with software-assisted error recovery option • 0003h = Switch

3.15.18 SP_LT_CTL

The port link time-out control CSR (SP_LT_CTL) is shown in Figure 3-87 and described in Table 3-81.

Figure 3-87. Port Link Timeout Control CSR—SP_LT_CTL

31 8	7 0
TIMEOUT_VALUE	Reserved
R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-81. Port Link Timeout Control CSR—SP_LT_CTL Field Descriptions

Bit	Field	Value	Description
31-8	TIMEOUT_VALUE	• FFFFFFh • 0FFFFFh • 00FFFFh • 000FFFh • 0000FFh • 00000Fh • 000001h • 000000h	Timeout interval value, is in the range provided by: • Shortest: SRV_CLK * TIMEOUT_VALUE * 3 • Longest: SRV_CLK * TIMEOUT_VALUE * 4 TIMEOUT_VALUE=0 disables the timer.
7-0	Reserved	00h	• These read-only bits return 0s when read.

3.15.19 SP_RT_CTL

The port response time-out control CSR (SP_RT_CTL) is shown in Figure 3-88 and described in Table 3-82.

Figure 3-88. Port Response Time-Out Control CSR—SP_RT_CTL

31 8	7 0
TIMEOUT_VALUE	Reserved
R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-82. Port Response Time-Out Control CSR—SP_RT_CTL Field Descriptions

Bit	Field	Value	Description
31-8	TIMEOUT_VALUE	000000h to FFFFFFh	Timeout value for all ports on the device. This timeout is for sending a packet to receiving the corresponding response packet. Max value represents 3 to 6 seconds. The timeout duration can be expressed as: Timeout = 15 × ((Prescale Value + 1) × DMA Clock Period × Timeout Value) where Prescale Value is set in PER_SET_CNTL (offset 0020h) and the Timeout Value is the decimal representation of this register value. For example, given a 400-MHz DMA, a Prescale Value of 4, and a Timeoout Value of FFFFFFh, the Timeoout duration would be: Timeout = 15 × ((4 + 1) × 2.5 ns × 16777216) = 3.15 s
7-0	Reserved	00h	These read-only bits return 0s when read.

3.15.20 SP_GEN_CTL

The port general control CSR (SP_GEN_CTL) is shown in Figure 3-89 and described in Table 3-83.

Figure 3-89. Port General Control CSR n—SP_GEN_CT

31	30	29	28 ... 0
HOST	MASTER_ENABLE	DISCOVERED	Reserved
R/W	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-83. Port General Control CSR n—SP_GEN_CT Field Descriptions

Bit	Field	Description
31	HOST	A host device is a device that is responsible for system exploration, initialization, and maintenance. Agent or slave devices are typically initialized by host devices. 0h = Agent or slave devices 1h = Host device
30	MASTER_ENABLE	The Master Enable bit controls whether or not a device is allowed to issue requests into the system. If the Master Enable is not set, the device may only respond to requests. 0h = Processing element cannot issue requests 1h = Processing element can issue requests
29	DISCOVERED	This device has been located by the processing element responsible for system configuration. 0h = The device has not been previously discovered 1h = The device has been discovered by another processing element
28-0	Reserved	• These read-only bits return 0s when read.

3.15.21 SPn_LM_REQ

Each of the four ports is supported by a register of this type. SP *n*_LM_REQ is shown in Figure 3-90 and described in Table 3-84.

Figure 3-90. Port Link Maintenance Request CSR n—SPn_LM_REQ

31	3	2	0
Reserved		COMMAND	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-84. Port Link Maintenance Request CSR n (SPn_LM_REQ) Field Description

Bit	Field	Value	Description
31-3	Reserved	0	These read-only bits return 0s when read.
2-0	COMMAND	000b-111b	Command to be sent in the link-request control symbol. If read, this field returns the last written value. • 0b011 = Reset • 0b100 = Input-status • Other = Reserved Note: Four link-request/reset-request control symbols are sent each time 0b011 is written to this field.

3.15.22 SPn_LM_RESP

Each of the four ports is supported by a register of this type. The port link maintenance response CSR *n* (SP *n*_LM_RESP) is shown in Figure 3-91 and described in Table 3-85.

Figure 3-91. Port Link Maintenance Response CSR n—SPn_LM_RESP

31	30	11 10	5 4	0
RESPONSE_VALID	Reserved	ACKID_STATUS	LINK_STATUS	
R	R	R	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-85. Port Link Maintenance Response CSR n—SPn_LM_RESP Field Descriptions

Bit	Field	Value	Description
31	RESPONSE_VALID		If the link-request causes a link-response, this bit indicates that the link-response has been received and the status fields are valid. If link-request does not cause a link-response, this bit indicates that the link-request has been transmitted. This bit automatically clears on read. Note: For link-response control symbols, this bit certifies the availability of data; it does not certify the correctness of the data.
30-10	Reserved	0	These read-only bits return 0s when read.
9-5	ACKID_STATUS	00000b-11111b	AckID status field from the link-response control symbol. The value of the next ackID expected by receiver.
4-0	LINK_STATUS	00000b-11111b	Link-Status field from the link-response control symbol • 0b00010 = Error • 0b00100 = Retry-stopped • 0b00101 = Error-stopped • 0b10000 = OK Other = Reserved

3.15.23 SPn_ACKID_STAT

Each of the four ports is supported by a register of this type. The port local ackID status CSR *n* (SP *n*_ACKID_STAT) is shown in Figure 3-92 and described in Table 3-86.

Figure 3-92. Port AckID Status Register CSR n

31	30 29	28 24	23 13	12 8	7 5	4 0
CLR_OUTSTD_ACK_ID	Reserved	Inbound_AckID	Reserved	Outstanding_Ackid	Reserved	Outbound_AckID
R/W			R			R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-86. Port AckID Status Register CSR n Field Descriptions

Bit	Name	R/W	Reset Value	Description
31	CLR_OUTSTD_ACK_ID	R/W	0b0	Clear outstanding ackIDs
30-29	Reserved	R		Reserved
28-24	Inbound_ackID	R	0b00000	Input port next expected ackID value. Bit 28 is valid only for long control symbols. Inbound_ackID indicates the expected value of the ackID field in the next packet to be received. Inbound_ackID may be written to synchronize the expected ackID with the ackID being used by the link partner. Inbound_ackID returns the instantaneous value when the physical layer processes the read request.
23-13	Reserved	R		Reserved
12-8	Outstanding_ackID	R	0x00000	Output port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol.
7-5	Reserved	R		Reserved
4-0	Outbound_ackID	R/W	0b00000	Output port next transmitted ackID value. Software writing this value can force retransmission of outstanding unacknowledged packets in order to manually implement error recovery.

3.15.24 SP(n)_CTL2

Each of the four ports is supported by a register of this type. The port control 2 register is shown in Figure 3-93 and described in Table 3-87:

Figure 3-93. Port n Control 2 CSR (Address offset 0xB154, 0xB174, 0xB194, 0xB1B4)

31 28	27	26	25	24	23	22	21
BAUD_SEL	BAUD_DISC	Reserved	GB_1p25	GB_1p25_EN	GB_2p5	GB_2p5_EN	GB_3p125
R	R	R	R	R/W	R	R/W	R

20	19	18	17	16	15 4	3	2	1	0
GB_3p125_EN	GB_5p0	GB_5p0_EN	GB_6p25	GB_6p25_EN	Rsvd	INACT_EN	D_SCRM_DIS	RTEC	RTEC_EN
R/W	R	R/W	R	R/W	R	R/W	R/W	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-87. Port n Control 2 CSR (Address offset 0xB154, 0xB174, 0xB194, 0xB1B4)

Bit	Name	R/W	Reset Value	Description
31-28	BAUD_SEL	R	0b0	Indicates the initialized baud rate of the port. 0b0000 = No rate selected 0b0001 = 1.25 GBaud 0b0010 = 2.5 GBaud 0b0011 = 3.125 GBaud 0b0100 = 5.0 GBaud 0b0101 = 6.25 GBaud 0b0110 = 0b1111 = Reserved
27	BAUD_DISC	R	0b0	Indicates whether automatic baud rate discovery is supported.
26	Reserved	R	0b0	Reserved
25	GB_1p25	R	0b1	Indicates whether port operation at 1.25 GBaud is supported.
24	GB_1p25_EN	R/W	0b0	Controls whether port operation at 1.25 GBaud is enabled.
23	GB_2p5	R	0b1	Indicates whether port operation at 2.5 GBaud is supported.
22	GB_2p5_EN	R/W	0b0	Controls whether port operation at 2.5 GBaud is enabled.
21	GB_3p125	R	0b1	Indicates whether port operation at 3.125 GBaud is supported.
20	GB_3p125_EN	R/W	0b0	Controls whether port operation at 3.125 GBaud is enabled.
19	GB_5p0	R	0b1	Indicates whether port operation at 5.0 GBaud is supported.
18	GB_5p0_EN	R/W	0b0	Controls whether port operation at 5.0 GBaud is enabled.
17	GB_6p25	R	0b1	Indicates whether port operation at 6.25 GBaud is supported.
16	GB_6p25_EN	R/W	0b0	Controls whether port operation at 6.25 GBaud is enabled.
15-4	Reserved	R	0b0	Reserved
3	INACT_EN	R/W	0b0	0 = Lanes assigned to the port but not used by the port have their outputs disabled 1 = Lanes assigned to the port but not used by the port have their transmitter and receiver enabled This RapidIO 2.1 standard bit is for test and debug use only.
2	D_SCRM_DIS	R/W	0b0	Data scrambling disable 0 = Transmit data scrambler and receive data descrambler are enabled 1 = Transmit data scrambler and receive data descrambler are disabled for control and packet data characters. Control symbol and packet data characters are neither scrambled in the transmitter before transmission nor descrambled in the receiver upon reception. The transmit scrambler remains enabled for the generation of pseudo-random data characters for the IDLE2 random data field. This bit is for test use only and must not be asserted during normal operation.
1	RTEC	R	0b0	Indicates whether the port can transmit commands to control the transmit emphasis in the connected port

Table 3-87. Port n Control 2 CSR (Address offset 0xB154, 0xB174, 0xB194, 0xB1B4) (continued)

Bit	Name	R/W	Reset Value	Description
0	RTEC_EN	R	0b0	Controls whether a port can adjust the transmit emphasis in the connected port. 0 = Remote transmit emphasis control is disabled 1 = Remote transmit emphasis control is enabled The port must not let this bit be set unless remote transmit emphasis control is supported and the link to which the port is connect is using Idle sequence 2 (IDLE2).

3.15.25 SPn_ERR_STAT

Each of the four ports is supported by a register of this type. The port error and status CSR *n* (SP *n*_ERR_STAT) is shown in Figure 3-94 and described in Table 3-88.

Figure 3-94. Port Error Status CSR n—SP(n)_ERR_STAT (Address Offset 0xB158, 0xB178, 0xB198, 0xB1B8)

31	30	29	28
IDLE2	IDLE2_EN	IDLE2_SEQ	Reserved
R	R	R	R

27	26	25	24	23	21	20
TXFC	Output Pkt Drop	Output Fld Enc	Output Degrd En	Reserved		Output Retry Enc
R	R/W	R/W	R/W			R/W

19	18	17	16	15	11	10	9
Output Retried	Output Retry Stp	Output Error Enc	Output Error Stp	Reserved		Input Retry Stp	Input Error Enc
R	R	R/W	R			R	R/W

8	7	5	4	3	2	1	0
Input Error Stp	Reserved		Port-Write Pnd	PORT_UNAVL	Port Error	Port OK	Port Uninitialized
R			R/W		R/W	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-88. Port Error Status CSR n Field Descriptions (Address Offset 0xB158, 0xB178, 0xB198, 0xB1B8)

Bit	Name	R/W	Reset Value	Description
31	IDLE2	R	0b0	Indicates whether the port supports idle sequence 2 for baud rate of less than 5.5 GBaud.
30	IDLE2_EN	R	0b0	Controls whether idle sequence 2 is enabled for baud rates of less than 5.5 Gbaud.
29	IDLE2_SEQ	R	0b0	Indicates which idle is active.
28	Reserved	R	0b0	Reserved
27	TXFC	R	0b0	Indicates the physical level flow control currently in use on the link.
26	Output Packet-dropped	R/W1c	0b0	Output port has discarded a packet. (switch devices only)
25	Output Failed-encountered	R/W1c	0b0	Output port has encountered a failed condition, meaning that the failed port error threshold has been reached in the Port *n* Error Rate Threshold register. Once set remains set until written with a logic 1 to clear.
24	Output Degraded-encountered	R/W1c	0b0	Output port has encountered a degraded condition, meaning that the degraded port error threshold has been reached in the Port *n* Error Rate Threshold register. Once set remains set until written with a logic 1 to clear.
23-21	Reserved	R	0b0	Reserved
20	Output Retry-encountered	R/W1c	0b0	Output port has encountered a retry condition. This bit is set when bit 18 is set. Once set, remains set until written with a logic 1 to clear.
19	Output Retried	R	0b0	Output port has received a packet-retry control symbol and can not make forward progress. This bit is set when bit 18 is set and is cleared when a packet-accepted or a packet-not-accepted control symbol is received (read-only).
18	Output Retry-stopped	R	0b0	Output port has received a packet-retry control symbol and is in the output retry-stopped state (read-only).
17	Output Error-encountered	R/W1c	0b0	Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 16 is set. Once set, remains set until written with a logic 1 to clear.
16	Output Error-stopped	R	0b0	Output is in the output error-stopped state (read-only).
15-11	Reserved	R	0b0	Reserved
10	Input Retry-stopped	R	0b0	Input port is in the input retry-stopped state (read-only).

Table 3-88. Port Error Status CSR n Field Descriptions (Address Offset 0xB158, 0xB178, 0xB198, 0xB1B8) (continued)

Bit	Name	R/W	Reset Value	Description
9	Input Error-encountered	R/W1c	0b0	Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit 8 is set. Once set, remains set until written with a logic 1 to clear.
8	Input Error-stopped	R	0b0	Input port is in the input error-stopped state (read-only).
7-5	Reserved			Reserved
4	Port-write Pending	R/W1c	0b0	Port has encountered a condition which required it to initiate a maintenance port-write operation This bit is only valid if the device is capable of issuing a maintenance port-write transaction. Once set remains set until written with a logic 1 to clear.
3	PORT_UNAVL	R	0b0	Indicates whether or not the port is available. The ports resources may have been merged with another port to support wider links.
2	Port Error	R/W1c	0b0	Port Error Inbound or outbound port has encountered an error from which the hardware was unable to recover (fatal error). Write 1 to clear this bit. This bit reports the failing of design to recover from following errors: - four link-request trials with receiving of link-response without an outstanding ackID - four link-request trials with time-out error for link-response.
1	Port OK	R	0b0	The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device (read-only).
0	Port Uninitialized	R	0b1	Input and output ports are not initialized. This bit and bit 1 are mutually exclusive (read-only).

3.15.26 SPn_CTL

Each of the four ports is supported by a register of this type. The port control CSR *n* (SP *n*_CTL) is shown in Figure 3-95 and described in Table 3-89.

Figure 3-95. Port Control Register CSR n—RIO_SP(n)_CTL

31–30	29–27	26–24	23	22
Port Width	Initialized Port Width	Port Width Override	Port Disable	Output Port Enable
R	R	R/W	R/W	R/W

21	20	19	18	17	16	15–14
Input Port Enable	Error Check Disable	Multicast Participant	Flow Control	Enum B	Flow arb	Over Pwidth2
R/W	R/W	R/W	R/W	R/W	R/W	R

13–12	11–4	3	2	1	0
Port Width2	Reserved	Stop Fail EN	Drop EN	Port Lockout	Port type
R	R	R/W	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-89. Port Control Register CSR n Field Descriptions

Bit	Name	R/W	Reset Source	Reset Value	Description
31-30	Port Width	R	VBUS_rst	0b00	Indicates port width modes supported by the port. This field is used in conjunction with the PORT_WIDTH2 field of this register. The bits of these two fields collectively indicate the port width modes supported by the port in addition to 1x mode which is supported by all ports. Bit 0: 0 = 4x mode not supported 1 = 4x mode supported Bit 1: 0 = 2x mode not supported 1 = 2x mode supported The value of this field is controlled by the settings of the RapidIO PLM Port {0..3} Path Control Register.PATH_CONFIGURATION and PATH_MODE.
29-27	Initialized Port Width	R	VBUS_rst	0b000	Width of the ports after initialized (read only): 0b000 - Single-lane port, lane 0 0b001 - Single-lane port, lane 2 0b010 - Four-lane port 0b011 - 0b111 — reserved
26-24	Port Width Override	R/W	VBUS_rst	0b000	Soft port configuration to override the hardware size: 0b000 - No override 0b001 — reserved 0b010 - Force single lane, lane 0 0b011 - Force single lane, lane 2 0b100 - 0b111 — reserved
23	Port Disable	R/W	VBUS_rst	0b0	Port disable: 0b0 - port receivers and drivers are enabled 0b1 - port receivers and drivers are disabled and are unable to receive or transmit to any packets or control symbols
22	Output Port Enable	R/W	VBUS_rst	0b0	Output port transmit enable: 0b0 - port is stopped and not enabled to issue any packets except to route or respond to I/O logical MAINTENANCE packets, depending upon the functionality of the processing element. Control symbols are not affected and are sent normally. 0b1 - port is enabled to issue any packets

Table 3-89. Port Control Register CSR n Field Descriptions (continued)

Bit	Name	R/W	Reset Source	Reset Value	Description
21	Input Port Enable	R/W	VBUS_rst	0b0	Input port receive enable: 0b0 - port is stopped and only enabled to route or respond I/O logical MAINTENANCE packets, depending upon the functionality of the processing element. Other packets generate packet-not-accepted control symbols to force an error condition to be signaled by the sending device. Control symbols are not affected and are received and handled normally. 0b1 - port is enabled to respond to any packet
20	Error Checking Disable	R/W	VBUS_rst	0b0	This bit disables all RapidIO transmission error checking 0b0 - Error checking and recovery is enabled 0b1 - Error checking and recovery is disabled Device behavior when error checking and recovery is disabled and an error condition occurs is undefined
19	Multicast-event Participant	R/W	VBUS_rst	0b0	Send incoming multicast-event control symbols to this port (multiple port devices only)
18	FLOW_CTRL	R/W	VBUS_rst	0b0	Enable flow control transactions
17	ENUM_B	R/W	VBUS_rst	0b0	Enumeration boundary bit
16	FLOW_ARB	R/W	VBUS_rst	0b0	Enable flow arbitration transactions
15-14	OVER_PWIDTH2	R	VBUS_rst	0b0	This bit field only applies to ports that support 8- and 16-lanes. For the SRIO-TEV2 this field is therefore read-only.
13-12	PORT_WIDTH2	R	VBUS_rst	0b0	This bit field only applies to ports that support 8- and 16-lanes. For SRIO-TEV2 this field is therefore read-only.
11-4	Reserved			0b0	Reserved
3	Stop on Port Failed-encountered Enable	R/W	VBUS_rst	0b0	When set, this bit causes the port to set the Port Error bit in the Port n Error and Status CSR and stop attempting to send packets to the connected device when the output failed-encountered bit is set. Packets are discarded if the Drop Packet Enable bit is set. When cleared the port continues to attempt to transmit packets to the connected device if the output failed-encountered bit is set.
2	Drop Packet Enable	R/W	VBUS_rst	0b0	When set this bit allows the output port to drop packets that are acknowledged with a packet-not-accepted control symbol when the error failed threshold is exceeded. If the port heals, and the current error rate falls below the failed threshold, the output no longer drops packets (switch devices only). When cleared, the output port continues to try to transmit packets that have been rejected due to transmission errors.
1	Port Lockout	R/W	VBUS_rst	0b0	Port Lockout: When cleared, this port is enabled to issue any packets When set, this port is stopped and is not enabled to issue or receive any packets; the input port can still follow the training procedure and can still send and respond to link-requests; all received packets return packet-not-accepted control symbols to force an error condition to be signaled by the sending device
0	Port Type	R	VBUS_rst	0b1	This indicates the port type, parallel or serial (read only) 0b0 - Parallel port 0b1 - Serial port

3.16 Error Management Registers

3.16.1 ERR_RPT_BH

Error Reporting Block Header register fields are shown in Figure 3-96 and explained in Table 3-90.

Figure 3-96. Error Reporting Block Header

31	16	15	0
EF_PTR		EF_ID	
R		R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-90. Error Reporting Block Header Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-16	EF_PTR	R	0x0000	Hard-wired pointer to the next block in the data structure. None exists.
15-0	EF_ID	R	0x0007	Hard-wired extended features ID

3.16.2 ERR_DET

Error Detect register fields are shown in Figure 3-97 and are explained in Table 3-91:

Figure 3-97. Error Detect—ERR_DET

31	30	29	28	27	26	25	24
IO ERR Rspns	Msg ERR Rspns	GSM ERR Rspns	ERR MSG Format	ILL Trans Decode	ILL Trans Trgt ERR	MSG REQ Timeout	PKT Rspns Timeout
R/W0c	R/W0c	R	R/W0c	R/W0c	R	R/W0c	R/W0c
23	22	21	8	7	6	5	0
Unsolicited Rspns	Unsupported Trans	Reserved		RX_CPPI Security	RX IO Access	Reserved	
R/W0c	R/W0c	R		R/W0c	R/W0c	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-91. Error Detect Field Descriptions

Bit	Name	R/W	Reset Value	Description
31	IO error response	R/W0c	0b0	Received a response of ERROR for an IO logical layer request. (end point device only) • Write 0 to clear • Write 1 for software debug purposes only
30	Message error response	R/W0c	0b0	Received a response of ERROR for an MSG logical layer request. (end point device only) • Write 0 to clear • Write 1 for software debug purposes only
29	GSM error response	R	0b0	Received a response of ERROR for a GSM logical layer request. (end point device only) NOT SUPPORTED — GSM logical layer not supported
28	Message Format Error	R/W0c	0b0	Received MESSAGE packet data payload with an invalid size or segment (MSG logical) (end point device only) • Write 0 to clear • Write 1 for software debug purposes only
27	Illegal transaction decode	R/W0c	0b0	Received illegal fields in the request/response packet for a supported transaction (IO/MSG/GSM logical) (switch or endpoint device) • Write 0 to clear • Write 1 for software debug purposes only
26	Illegal transaction target error	R	0b0	Received a packet that contained a destination ID that is not defined for this end point. NOT SUPPORTED.
25	Message Request Time-out	R/W0c	0b0	A required message request has not been received within the specified time-out interval (MSG logical) (end point device only) • Write 0 to clear • Write 1 for software debug purposes only
24	Packet Response Time-out	R/W0c	0b0	A required response has not been received within the specified timeout interval (IO/MSG/GSM logical) (end point device only) • Write 0 to clear • Write 1 for software debug purposes only
23	Unsolicited Response	R/W0c	0b0	An unsolicited or unexpected response packet was received (IO/MSG/GSM logical; only maintenance response for switches) (switch or endpoint device) • Write 0 to clear • Write 1 for software debug purposes only

Table 3-91. Error Detect Field Descriptions (continued)

Bit	Name	R/W	Reset Value	Description
22	Unsupported Transaction	R/W0c	0b0	A transaction is received that is not supported in the destination operations CAR (IO/MSG/GSM logical; only maintenance port-write for switches) (switch or endpoint device) • Write 0 to clear • Write 1 for software debug purposes only
21-15	Reserved			Reserved
14	Data Streaming PDU Length Error	R/W0c	0b0	The length of a reassembled PDU differs from the PDU length carried in the end data-streaming segment packet header (end point device only).
13	Short Data Streaming Segment	R/W0c	0b0	Received a start or continuation data streaming segment with a payload size less than that the MTU (end point device only).
12	Long Data Streaming Segment	R/W0c	0b0	Received a data streaming segment with a payload size greater than the MTU (end point device only).
11	Open Existing Data Streaming Context	R/W0c	0b0	A start or single data streaming segment was received for an already open segmentation context (end point device only).
10	Missing Data Streaming Context	R/W0c	0b0	A continuation or end data streaming segment was received for a closed or non-existent segmentation context (end point device only).
9	No Context Available for Type9 Traffic Causing the Packet to be Dropped	R/W0c	0b0	No context available for type9 traffic, causing the packet to be dropped.
8	Reserved			Reserved
7	RX CPPI Security Violation	R/W0c	0b0	Access to one of the RX queues was blocked • Write 0 to clear • Write 1 for software debug purposes only
6	RX I/O DMA Access Error	R/W0c	0b0	DMA access to MAU was blocked • Write 0 to clear • Write 1 for software debug purposes only
5-0	Reserved			Reserved

3.16.3 ERR_EN

Error Enable register is shown in Figure 3-98 and described in Table 3-92:

Figure 3-98. Logical/Transport Layer Error Enable CSR—ERR_EN (Address Offset 0xC00C)

31	30	29	28	27
IO ERR Resp Enable	MSG ERR Resp Enable	GSM ERR Resp Enable	ERR MSG Format Enable	ILL Trans Decode Enable
R/W	R/W	R/W	R/W	R/W

26	25	24	23
ILL Trans Target Err Enable	MSG Req Timeout Enable	PKT Resp Timeout Enable	Unsolicited Resp Enable
R/W	R/W	R/W	R/W

22	21 15	14	13	12
Unsupported Trans Enable	Reserved	Data Str PDU Len Err En	Short Data Str Seg Err En	Long Data Str Seg Err En
R/W	R	R/W	R/W	R/W

11	10	9	8	7	6	5 0
Open Ex Data Str Con Err En	Missing Ex Data Str Con Err En	RXU Pkt Dropped No Con Type9	Rsvd	RX CPPI Sec Err En	RX I/O Sec Err En	Reserved
R/W	R/W	R/W	R	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-92. Logical//Transport Layer Error Enable CSR—ERR_EN Field Descriptions

Bit	Name	R/W	Reset Value	Description
31	IO error response enable	R/W	0b0	Enable reporting of an IO error response. Save and lock original request transaction capture information in all Logical/Transport Layer Capture CSRs. (end point device only)
30	Message error response enable	R/W	0b0	Enable reporting of a message error response. Save and lock transaction capture information in all Logical/Transport Layer Capture CSRs. (end point device only)
29	GSM error response enable	R/W	0b0	Enable reporting of a GSM error response. Save and lock original request address in Logical/Transport Layer Address Capture CSRs. Save and lock transaction capture information in all Logical/Transport Layer Device ID and Control CSRs. (end point device only)
28	Message Format Error enable	R/W	0b0	Enable reporting of a message format error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (end point device only)
27	Illegal transaction decode enable	R/W	0b0	Enable reporting of an illegal transaction decode error Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (switch or end-point device)
26	Illegal transaction target error enable	R/W	0b0	Enable reporting of an illegal transaction target error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (end point device only) NOT APPLICABLE – PACKET DESTROYED BEFORE REACHING LOGICAL LAYER.
25	Message Request time-out enable	R/W	0b0	Enable reporting of a Message Request time-out error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs for the last Message request segment packet received. (end point device only)
24	Packet Response Time-out error enable	R/W	0b0	Enable reporting of a packet response time-out error. Save and lock original request address in Logical/Transport Layer Address Capture CSRs. Save and lock original request Destination ID in Logical/Transport Layer Device ID Capture CSR. (end point device only)
23	Unsolicited Response error enable	R/W	0b0	Enable reporting of an unsolicited response error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (switch or end-point device)

Table 3-92. Logical//Transport Layer Error Enable CSR—ERR_EN Field Descriptions (continued)

Bit	Name	R/W	Reset Value	Description
22	Unsupported Transaction error enable	R/W	0b0	Enable reporting of an unsupported transaction error. Save and lock transaction capture information in Logical/Transport Layer Device ID and Control Capture CSRs. (switch or end-point device)
21-15	Reserved			Reserved
14	Data streaming PDU length error enable	R/W	0b0	Enable reporting of a reassembled PDU length that differs from the PDU length carried in the end data streaming segment packet header. Save and lock capture information in the appropriate Logical/Transport Layer Capture CSRs. (end point device only)
13	Short data streaming segment error enable	R/W	0b0	Enable reporting of a start or continuation data streaming segment received with a payload size less that the MTU. Save and lock capture information in the appropriate Logical/Transport Layer Capture CSRs. (end point device only)
12	Long data streaming segment error enable	R/W	0b0	Enable reporting of a any data streaming segment received with a payload size greater then the MTU. Save and lock capture information in the appropriate Logical/Transport Layer Capture CSRs. (end point device only)
11	Open existing data streaming context error enable	R/W	0b0	Enable reporting of a start or single data streaming segment received for an already open segmentation context. Save and lock capture information in the appropriate Logical/Transport Layer Capture CSRs. (end point device only)
10	Missing data streaming context error enable	R/W	0b0	Enable reporting of a continuation or end data streaming segment received for a closed or non-existent segmentation context. Save and lock capture information in the appropriate Logical/Transport Layer Capture CSRs. (end point device only)
9	RXU Pkt Dropped as no Context available for type9 traffic causing the packet to be dropped	R/W	0b0	No Context available for type9 traffic causing the packet to be dropped
8	Reserved			Reserved
7	RX CPPI Security error enable	R/W	0b0	Enable reporting of attempt at unauthorized access to a RX queue. Save and Lock capture information in appropriate Logical/Transport Layer Capture CSRs.
6	RX I/O Security error enable	R/W	0b0	Enable reporting of attempt at unauthorized memory location access. Save and Lock capture information in appropriate Logical/Transport Layer Capture CSRs.
5-0	Reserved			Reserved

3.16.4 H_ADDR_CAPT

Logical/Transport Layer High Address Capture CSR is shown in Figure 3-99 and explained in Table 3-93:

Figure 3-99. High Address Capture CSR—H_ADDR_CAPT

31	0
Address [63-32]	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-93. High Address Capture CSR Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-0	Address[63-32]	R/W*	All 0's	Most Significant 32 bits of the address associated with the error (only valid for devices supporting 66- and 50-bit addresses)

Locked when error detected and enable bit set. Unlocked when software writes all 0s to RIO_ERR_DET (0x2008). The bits are writeable by software for debug purposes.

3.16.5 ADDR_CAPT

Logical/Transport Layer Address Capture CSR is shown in Figure 3-100 and explained in Table 3-94:

Figure 3-100. Address Capture CSR

31	3	2	1	0
Address [31-2]		Rsvd	Xamsbs	
R/W			R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-94. Address Capture CSR Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-3	Address[31-3]	R/W*	All 0's	Least Significant 29 bits of the address associated with the error
2	Reserved		0b0	Reserved
1-0	Xamsbs	R/W*	0b00	Extended address bits of the address associated with the error.

Locked when error detected and enable bit set. Unlocked when software writes all 0s to RIO_ERR_DET (0x2008). The bits are writeable by software for debug purposes.

3.16.6 ID_CAPT

Logical/Transport Layer Device ID Capture CSR is shown in Figure 3-101 and explained in Table 3-95:

Figure 3-101. Device ID Capture CSR—ID_CAPT

31 24	23 16	15 8	7 0
MSB DESTID	DESTID	MSB SOUCEID	SOURCEID
R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-95. Device ID Capture CSR

Bit	Name	R/W	Reset Value	Description
31-24	MSB destinationID	R/W*	0x00	Most significant byte of the destinationID associated with the error (large transport systems only)
23-16	destinationID	R/W*	0x00	The destinationID associated with the error
15-8	MSB sourceID	R/W*	0x00	Most significant byte of the sourceID associated with the error (large transport systems only)
7-0	sourceID	R/W*	0x00	The sourceID associated with the error

Locked when error detected and enable bit set. Unlocked when software writes all 0s to RIO_ERR_DET (0x2008). The bits are writeable by software for debug purposes.

3.16.7 CTRL_CAPT

Logical/Transport Layer Control Capture CSR is shown in Figure 3-102 and explained in Table 3-96:

Figure 3-102. Control Capture CSR—CTRL_CAPT

31 28	27 24	23 16	15 0
FTYPE	TTYPE	MSGINFO	Imp Specific
R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-96. Control Capture CSR

Bit	Name	R/W	Reset Value	Description
31-28	ftype	R/W*	0x0	Format type associated with the error
27-24	ttype	R/W*	0x0	Transaction type associated with the error
23-16	msg info	R/W*	0x00	letter, mbox, and msgseg for the last message request received for the mailbox that had an error (message errors only) For multi-segment message: Bit#16-19 : SegmentID Bit#20-21 : mbox Bit#22-23 : Letter For single segment message: Bit#16-21 : mbox Bit#22-23 : Letter For Type 11 packets, Bit#23-22: Letter Bit#21-20: Mbox Bit#19-16: msgseg
15-0	Implementation specific	R/W*	0x0000	Implementation-specific information associated with the error

Locked when error detected and enable bit set. Unlocked when software writes all 0s to RIO_ERR_DET (0x2008). The bits are writeable by software for debug purposes.

3.16.8 PW_TGT_ID

Logical/Transport Layer Port Write Target Device ID CSR is shown in Figure 3-103 and explained in Table 3-97:

Figure 3-103. Port Write Target Device ID CSR—PW_TGT_ID

31 24	23 16	15	14 0
DeviceID_MSB	DeviceID	ID_Large	Reserved
R/W	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-97. Port Write Target Device ID CSR

Bit	Name	R/W	Reset Value	Description
31-24	deviceID_msb	R/W	0x00	This is the most significant byte of the port-write target deviceID (large transport systems only)
23-16	deviceID	R/W	0x00	This is the port-write target deviceID
15	Id_large	R/W	0b0	0b0 — use 8b deviceID for port-write operation 0b1 — use 16b deviceID for port-write operation
14-0	Reserved			Reserved

3.16.9 SPx_ERR_DET

Port x Error Detect register is shown in Figure 3-104 and explained in Table 3-98:

Figure 3-104. Port n Error Detect CSR (Address offset 0xC040, 0xC080, 0xC0C0, 0xC100)

31	30 23	22	21	20	19	18
Implementation specific error	Reserved	CS_CRC_ERR	CS_ILL_ID	CS_NOT_ACC	PKT_ILL_ACKID	PKT_CRC_ERR
R/W	R	R/W	R/W	R/W	R/W	R/W

17	16 15	14 13 6	5	4
PKT_ILL_SIZE	Reserved	DSCRAM_LOS	Non-outstanding ackID	Protocol error
R/W	RR/W	R/W	R/W	R/W

3	2	1	0
Reserved	Delineation error	Unsolicited acknowledge control symbol	Link time-out
R	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-98. Port n Error Detect CSR (Address offset 0xC040, 0xC080, 0xC0C0, 0xC100)

Bit	Name	R/W	Reset Value	Description
31	Implementation specific error	R/W0c	0b0	An implementation-specific error has been detected. It covers: illegal tt field (tt>01), unmapped DestID field (only valid when bit 26 of 0x120004 is 0), illegal/reserved field encoding of maintenance packet, fatal input/output error (bit 2 of 0x1158), Output packet dropped, too many retries, not supported transaction. Write 0 to clear Write 1 for software debug purposes only
30-23	Reserved	R	0b0	Reserved
22	CS_CRC_ERR	R/W0c	0b0	Received control signal with bad CRC
21	CS_ILL_ID	R/W0c	0b0	Received an acknowledge control symbol with an unexpected ackID (packet-accepted or packet_retry)
20	CS_NOT_ACC	R/W0c	0b0	Received packet-not-accepted control symbol
19	PKT_ILL_ACKID	R/W0c	0b0	Received packet with unexpected ackID
18	PKT_CRC_ERR	R/W0c	0b0	Received packet with bad CRC
17	PKT_ILL_SIZE	R/W0c	0b0	Received packet exceeds 276 bytes
16-15	Reserved			Reserved
14	DSCRAM_LOS	R/W0c	0b0	When control symbol and packet data is being scrambled before transmission, the loss of receiver de-scrambler synchronization. No information is captured in RapidIO Port {0..3} Packet/Control Symbol Error Capture CSR 0,1,2 and 3.
13-6	Reserved	R	4h0	Reserved
5	Non-outstanding ackID	R/W0c	0b0	Link_response received with an ackID that is not outstanding. The capture registers do not have valid information during this error detection. Write 0 to clear Write 1 for software debug purposes only
4	Protocol error	R/W0c	0b0	An unexpected packet or control symbol was received Write 0 to clear Write 1 for software debug purposes only
3	Reserved			Reserved
2	Delineation error	R	0b0	Received unaligned /SC/ or /PD/ or undefined code-group (serial). The capture registers do not have valid information during this error detection. Write 0 to clear Write 1 for software debug purposes only

Table 3-98. Port n Error Detect CSR (Address offset 0xC040, 0xC080, 0xC0C0, 0xC100) (continued)

Bit	Name	R/W	Reset Value	Description
1	Unsolicited acknowledge control symbol	R/W0c	0b0	An unexpected acknowledge control symbol was received Write 0 to clear Write 1 for software debug purposes only
0	Link time-out	R/W0c	0b0	An acknowledge or link-response control symbol is not received within the specified time-out interval. The capture registers do not have valid information during this error detection. Write 0 to clear Write 1 for software debug purposes only

3.16.10 SPx_RATE_EN

Port Error Enable register is shown in Figure 3-105 and explained in Table 3-99.

Figure 3-105. Port Error Enable—SP(n)_RATE_EN

31	30	24	23
EN Imp Specific	Reserved		Invalid
R	R		R

22	21	20	19
Corrupt Cntl Sym Enable	Cntl Sym Unexpected AckID En	Rcved Pkt-not-Accpt En	PKT Unexpected AckID En
R	R	R	R

18	17	16 6	5
Rcved Pkt w/ bad CRC En	Rcved PKT Over 276B En	Reserved	Non-Outstanding AckID En
R	R	R	R

4	3	2	1	0
Protocol Error En	Invalid	Invalid	Unsolicited Ack Cntl Sym En	Link Time-out En
R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-99. Port Error Enable Field Descriptions

Bit	Name	R/W	Reset Value	Description
31	Implementation-specific error enable	R/W	0b0	Enable error rate counting of implementation specific errors
30-24	Reserved			Reserved
23	Received S-bit error enable	R/W	0b0	Enable error rate counting of a packet/control symbol with an S-bit parity error
22	Received control symbol with bad CRC enable	R/W	0b0	Enable error rate counting of a corrupt control symbol
21	Received out-of-sequence acknowledge control symbol enable	R/W	0b0	Enable error rate counting of an acknowledge control symbol with an unexpected ackID
20	Received packet-not-accepted control symbol enable	R/W	0b0	Enable error rate counting of received packet-not-accepted control symbols
19	Received packet with unexpected ackID enable	R/W	0b0	Enable error rate counting of packet with unexpected ackID value - out-of-sequence ackID
18	Received packet with Bad CRC enable	R/W	0b0	Enable error rate counting of packet with a bad CRC value
17	Received packet exceeds 276 Bytes enable	R/W	0b0	Enable error rate counting of packet which exceeds the maximum allowed size
16-6	Reserved			Reserved
5	Non-outstanding ackID enable	R/W	0b0	Enable error rate counting of link-responses received with an ackID that is not outstanding
4	Protocol error enable	R/W	0b0	Enable error rate counting of protocol errors
3	Frame toggle edge error enable	R/W	0b0	Enable error rate counting of frame toggle edge errors
2	Delineation error	R/W	0b0	Enable error rate counting of delineation errors
1	Unsolicited acknowledge control symbol	R/W	0b0	Enable error rate counting of unsolicited acknowledge control symbol errors
0	Link time-out	R/W	0b0	Enable error rate counting of link time-out errors

3.16.11 SPx_ERR_ATTR_CAPT_DBG0

Port Error Attribute Capture Debug0 register is shown in Figure 3-106 and explained in Table 3-100.

Figure 3-106. Port Error Attribute Capture Debug0—RIO_SP(n)_ERR_ATTR_CAPT_DBG0

31 30	29	28 24	23 4	3 1	0
INFO TYPE	Reserved	Error TYPE	Imp Specific	Reserved	Capture Valid Info
R/W	R	R/W	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-100. Port Error Attribute Capture Debug0 Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-30	Info type	R/W	0b00	Type of information logged 00 - packet 01 - control symbol (only error capture register 0 is valid) 10 - implementation-specific (capture register contents are implementation-specific) 11 - undefined (S-bit error), capture as if a packet (parallel physical layer only)
29	Reserved			Reserved
28-24	Error type	R/W	0x00	Encoded value of captured error bit in the Port *n* Error Detect Register
23-4	Implementation-dependent	R/W	All 0s	Implementation-dependent error information
3-1	Reserved			Reserved
0	Capture valid info	R/W0c	0b0	This bit is set by hardware to indicate that the packet and control symbol capture registers contain valid information. For control symbols, only capture register 0 will contain meaningful information. The software writes 0 to clear this bit and subsequently unlock all capture registers of port n.

3.16.12 SPx_ERR_CAPT_0_DBG1

Port n Packet/Control symbol Error Capture CSR register is shown in Figure 3-107 and explained in Table 3-101.

Figure 3-107. Packet/Control symbol Error Capture CSR—SP(n)_ERR_CAPT_0_DBG1

31	0
Capture 0	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-101. Packet/Control symbol Error Capture Field Descriptions CSR

Bit	Name	R/W	Reset Value	Description
31-0	Capture 0	R/W	All 0's	Control character and control symbol or 0 to 3 bytes of packet header

3.16.13 SPx_ERR_CAPT_1_DBG2

Port n Packet Error Capture 1 CSR register is shown in Figure 3-108 and explained in Table 3-102.

Figure 3-108. Packet Error Capture 1 CSR —RIO_SP(n)_ERR_CAPT_1_DBG2

31	0
Capture 1	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-102. Packet Error Capture 1 Debug Field Descriptions CSR

Bit	Name	R/W	Reset Value	Description
31-0	Capture 1	R/W	All 0's	4 to 7 bytes of packet header

3.16.14 SPx_ERR_CAPT_2_DBG3

Port n Packet Error Capture CSR register is shown in Figure 3-109 and explained in Table 3-103.

Figure 3-109. Packet Error Capture CSR—RIO_SP(n)_ERR_CAPT_2_DBG3

31	0
Capture 2	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-103. Packet Error Capture CSR Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-0	Capture 2	R/W	All 0's	8 to 11 bytes of packet header

The bits are writeable by software for debug purposes.

3.16.15 SPx_ERR_CAPT_3_DBG4

Port n Packet Error Capture CSR3 register is shown in Figure 3-110 and explained in Table 3-104.

•

Figure 3-110. Packet Error Capture CSR3—RIO_SP(n)_ERR_CAPT_3_DBG4

31	0
Capture 3	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-104. Packet Error Capture CSR3 Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-0	Capture 3	R/W	All 0's	12 to 15 bytes of packet header

3.16.16 SPx_ERR_RATE

Port n Error Rate CSR register is shown in Figure 3-111 and explained in Table 3-105.

Figure 3-111. Error Rate CSR—RIO_SP(n)_ERR_RATE

31 24	23 18	17 16	15 8	7 0
Error Rate Bias	Reserved	Error Rate Recovery	Peak Error Rate	Error Rate Counter
R/W	R	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-105. Error Rate CSR Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-24	Error Rate Bias	R/W	0x80	The error rate bias value. Freq bus - 156.25 MHz • 00 - do not decrement error rate counter • 01 - decrement every 0.84ms • 02 - decrement every 6.72ms • 04 - decrement every 53.77ms • 08 - decrement every 0.86s • 10 - decrement every 6.87s • 20 - decrement every 55s • 40 - decrement every 879s • 80 - decrement every 7037s • FF - decrement every 0.82us for debug only • Other — reserved
23-18	Reserved			Reserved
17-16	Error Rate Recovery	R/W	0b00	Error Rate Recovery This bits limit the incrementing of the error rate counter above the failed threshold trigger • 00 - 2 error above • 01 - 4 error above • 10 - 16 errors above • 11 - no limit incrementing the error rate counter
15-8	Peak Error Rate	R/W	0x00	This field contains the peak value attained by the error rate counter.
7-0	Error Rate Counter	R/W	0x00	These bits maintain a count of the number of transmission errors detected by the port. This number is decremented by the error rate bias mechanism. The counter cannot overflow or underflow and continue to increment or decrement as defined, even if thresholds are met. The software may reset this counter. If the value of the counter equals the error rate threshold trigger register, the error is reported.

3.16.17 SPx_ERR_THRESH

Port n Error Rate Threshold CSR register is shown in Figure 3-112 and explained in Table 3-106.

Figure 3-112. Error Rate Threshold CSR—RIO_SP(n)_ERR_THRESH

31 24	23 16	15 0
Error Rate Failed Threshold	Error Rate Degraded Thres	Reserved
R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-106. Error Rate Threshold CSR Field Descriptions

Bit	Name	R/W	Reset Value	Description
31-24	Error Rate Failed Threshold Trigger	R/W	0xFF	These bits provide the threshold value for reporting an error condition due to a possibly broken link. • 0x00 - Disable the error rate register • 0x01 - Set the error reporting threshold to 1 • 0x02 - Set the error reporting threshold to 2 • ... • 0xFF - Set the error reporting threshold to 255
23-16	Error Rate Degraded Threshold Trigger	R/W	0xFF	These bits provide the threshold value for reporting an error condition due to a degrading link. • 0x00 - Disable the error rate register • 0x01 - Set the error reporting threshold to 1 • 0x02 - Set the error reporting threshold to 2 • ... • 0xFF - Set the error reporting threshold to 255
15-0	Reserved			Reserved

3.17 Extended Features Block

3.17.1 LANEn_STAT0

Each of the four ports is supported by a register of this type. LANE *n*_STAT0 is shown in Figure 3-113 and described in Table 3-107.

Figure 3-113. Lane 0 Status 0 CSR—LANEn_STAT0 (Address Offset 0x0E010, 0x0E030, 0x0E050, 0x0E070)

31–24	23–20	19	18	17–16	15	14–13
PORT_NUM	LANE_NUM	TX_TYPE	TX_MODE	RX_TYPE	RX_INV	RX_TRN	RX_SYNC
R	R	R	R	R	R	R	R

12	11–8	7–6	5–4	3–2	1–0
RX_RDY	ERR_CNT	CHG_SYNC	CHG_TRN	Reserved	STAT1	STAT2_7
R	R	R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-107. Lane 0 Status 0 CSR—LANEn_STAT0 Field Descriptions (Address Offset 0x0E010, 0x0E030, 0x0E050, 0x0E070)

Bit	Name	R/W	Reset Value	Description
31-24	PORT_NUM	R	Port0 = 8'h0 Port1 = 8'h1 Port2 = 8'h2 Port3 = 8'h3	Port number associated with this lane. This varies depending on: The path configuration specified in RapidIO PLM Port {0..3} Path Control Register. PATH_CONFIGURATION. The lowest numbered lane of the path used in a port is the port number as follows: • Lane A: port 0 • Lane B: port 1 • Lane C: port 2 • Lane D: port 3 For example, in the 2x+1x+1x configuration, the 2x port uses lanes A and B and is numbered port 0; the 1x ports use C and D respectively and so are numbered 2 and 3, respectively.
23-20	LANE_NUM	R	4'b0	Lane within the port that this lane supports. This varies depending on: • The path configuration specified in RapidIO PLM Port {0..3} Path Control Register. PATH_CONFIGURATION • The lane within the path independent of lane swapping. The numbering of the lanes is: • For 4 1x ports, the lanes A, B, C, and D are numbered 0, 1, 2 and 3 respectively • For a 2x using A&B as the 2x port, the 2x A and B port is numbered 0 and C and D are numbered 2 and 3, respectively • For a 2x using C&D as the 2x port, the 2x C and D port is numbered 2 and A and B are numbered 0 and 1, respectively • For 1 4x port, all lanes are numbered 0
19	TX_TYPE	RE	0	Transmitter Type • 0 = Short reach • 1 = Long reach
18	TX_MODE	R	1'b0	Transmitter Mode Current operating mode of the transmitter: • 0 = Short reach • 1 = Long reach This field's value is set by srio_laneN_tx_mode.
17-16	RX_TYPE	RE	00	Receiver Type • 0b00 = Short run • 0b01 = Medium run • 0b10 = Long run • 0b11 = Reserved

Table 3-107. Lane 0 Status 0 CSR—LANEn_STAT0 Field Descriptions (Address Offset 0x0E010, 0x0E030, 0x0E050, 0x0E070) (continued)

Bit	Name	R/W	Reset Value	Description
15	RX_INV	R	0	Receiver Input Inverted This bit indicates whether the lane receiver has detected that the polarity of its input signal is inverted and has inverted its receiver input to correct the polarity. • 0 = Receiver input not inverted • 1 = Receiver input inverted Note: SRIO-TEV2 does not perform automatic polarity detection and correction. This field is always zero. Manual polarity inversion may be configured in RapidIO PLM Port {0..3} Lane Polarity Control Register.RXn_POL.
14	RX_TRN	R	1	Receiver Trained Always 1, as the SRIO-TEV2 does not support adaptive equalization.
13	RX_SYNC	R	0	Indicates whether the receiver has achieved lane synchronization. • 0 = Not in sync • 1 = Receiver has achieved lane_sync according to the specification
12	RX_RDY	R	0	Receiver Ready. Set to 1 whenever adaptive equalization adjustment complete, and RX_SYNC is achieved. Always the same value as RX_SYNC, as the SRIO-TEV2 does not support adaptive equalization.
11-8	ERR_CNT	RC	Undefined	Count of the number of 8b/10b decoding errors that have occurred since the last time this register was read. The error counter hits a limit at 15; in other words, it does not roll over. This value should not be verified after reset
7	CHG_SYNC	RC	0	Change in the RX_SYNC state since the last time this register was read • 0 = No change in RX_SYNC state • 1 = The RX_SYNC state changed since the last time this register was read.
6	CHG_TRN	R	0	Change in the RX_TRN value since the last time this register was read.
5-4	Reserved	R	0	
3	STAT1	RE	1	Indicates the existence of the STAT1 register. The STAT1 register is only useful if the application is using IDLE2/long control symbol/ >5.5 Gbaud lane rates.
2-0	STAT2_7	RE	0	Indicates the existence of the STAT2 to STAT7 registers. Do not allow the implementation of STAT2 to STAT7.

3.17.2 LANEn Status 1 CSR (LANE n_STAT1)

Each of the four ports is supported by a register of this type. LANE *n*_STAT1 is shown in Figure 3-114 and described in Table 3-108.

Figure 3-114. Lane 0 Status 1 CSR—LANEn_STAT1 (Address Offset 0x0E014, 0x0E034, 0x0E054, 0x0E074)

31	30	29	28	27	26 24	23 20	19 18	17 16	15	14 0
IDLE2	INFO _OK	CHG	IMPL _SPEC	LP_RX _TRN	LP_WIDTH	LP_LANE_NUM	LP_TAP _M1	LP_TAP _P1	LP_SCRM	Reserved
R/W	R	R	R	R	R	R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-108. Lane 0 Status 1 CSR—LANE n_STAT1 Field Descriptions (Address Offset 0x0E014, 0x0E034, 0x0E054, 0x0E074)

Bit	Name	R/W	Reset Value	Description
31	IDLE2	R/W1C	0b0	Indicates whether IDLE2 sequence is being received. • 0 = No IDLE2 sequence has been received since the last time the bit was cleared. • 1 = IDLE2 sequence has been received at some time since the bit was cleared.
30	INFO_OK	R	0b0	Indicates whether the IDLE2 information latched in this register is current. • 0 = Information is stale. Set when errors have been detected in the last IDLE2 sequence, or the lane_sync signal has flickered since the last CS marker and CS field were received. • 1 = Information is current. Information is from the last IDLE2 sequence received, no errors have been detected, and the lane_sync signal has remained asserted since the last CS marker and CS field were received.
29	CHG	RC	0b0	Indicates whether the value of any of the other bits in this register have changed since the last time the register was read. 0 = Field values are unchanged since the last read 1 = At least one field value has changed since the last time this register was read
28	IMPL_SPEC	R	0b0	Implementation-specific meaning, according to the link partners definition.
27	LP_RX_TRN	R	0b0	Indicates whether the link partners receiver is trained • 0 = Receiver not trained • 1 = Receiver is trained
26-24	LP_WIDTH	R	0b0	Link Partner Port Width • 0b000 = 1 lane • 0b001 = 2 lanes • 0b010 = 4 lanes • 0b011 = 8 lanes • 0b100 = 16 lanes • 0b101-0b111 = Reserved
23-20	LP_LANE_NUM	R	0b0	Link Partner Lane Number status The lane within the link-partners port that this lane is associated with. • 0b0000 = Lane 0 • 0b0001 = Lane 1 • ... • 0b1111 = Lane 15
19-18	LP_TAP_M1	R	0b0	Link Partner Tap Minus 1 status • 0b00 = Tap(-1) not implemented • 0b01 = Tap(-1) at minimum emphasis • 0b10 = Tap(-1) at maximum emphasis • 0b11 = Tap(-1) at intermediate emphasis setting

Table 3-108. Lane 0 Status 1 CSR—LANE n_STAT1 Field Descriptions (Address Offset 0x0E014, 0x0E034, 0x0E054, 0x0E074) (continued)

Bit	Name	R/W	Reset Value	Description
17-16	LP_TAP_P1	R	0b0	Link Partner Tap Plus 1 status • 0b00 = Tap(+1) not implemented • 0b01 = Tap(+1) at minimum emphasis • 0b10 = Tap(+1) at maximum emphasis • 0b11 = Tap(+1) at intermediate emphasis setting
15	LP_SCRM	R	0b0	Link Partner Data Scrambling and De-Scrambling setting • 0 = Scrambling and de-scrambling are disabled • 1 = Scrambling and de-scrambling are enabled
14-0	Reserved	R	0b0	Reserved

3.18 Physical Layer Implementation-Specific Registers

Registers starting from offset 0x1B000h are physical layer registers.

3.18.1 PLM IDT-Specific Block Header Register (PLM_BH)

This register identifies the following registers as the IDT PLM register block.

Figure 3-115. PLM Block Header (Address offset 0x1B000)—PLM_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-109. PLM Block Header (Address offset 0x1B000)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x0103	Pointer to the next IDT block, in units of 0x100 bytes.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.2 PLM Port(n) Implementation-Specific Control Register (PLM_SP(n)_IMP_SPEC_CTL)

This per-port register provides implementation-specific control of the port.

Figure 3-116. PLM Port(n) Implementation-Specific Control Register (Address offset 0x1B080, 0x1B100, 0x1B180, 0x1B200)—PLM_SP(n)_IMP_SPEC_CTL

31	30	29	28	27	26	25	24
PAYL_CAP	Rsvd	USE_IDLE1	DLB_EN	Reserved	FORCE_REINIT	SOFT_RST_PORT	TX_BYPASS
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W

23	22	21	20	19 18	17 16	15 0
LLB_EN	Reserved	PORT_SELF_RST	SELF_RST	SWAP_TX	SWAP_RX	DLT_THRESH
R/W	R	R/W	R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-110. PLM Port(n) Implementation Specific Control Register (Address offset 0x1B080, 0x1B100, 0x1B180, 0x1B200)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	PAYL_CAP	R/W	VBUS_rst	0b0	Payload capture. Setting this bit changes the behavior of the RapidIO Port {0..3}Packet/Control Symbol Error Capture CSR 0 and subsequent capture registers.
30	Reserved	R		0b0	Reserved
29	USE_IDLE1	R/W	VBUS_rst	0b0	This bit allows the use of the IDLE1 sequence above 5.5 Gbaud.
28	DLB_EN	R/W	VBUS_rst	0b0	Digital equipment loopback mode for port
27	Reserved	R		0b0	Reserved
26	FORCE_REINIT	R/W1s	VBUS_rst	0b0	Force link re-initialization process
25	SOFT_RST_PORT	R/W	VBUS_rst	0b0	Software reset control for the this port.
24	TX_BYPASS	R/W	VBUS_rst	0b0	Bypass the transmitter clock crossing FIFO
23	LLB_EN	R/W	VBUS_rst	0b0	Line loopback mode for port
22	Reserved	R		0b0	Reserved
21	PORT_SELF_RST	R/W	VBUS_rst	0b0	If SELF_RST is 0, PORT_SELF_RST determines the port's behavior when the port receives a reset request.
20	SELF_RST	R/W	VBUS_rst	0b0	SELF_RST determines the port's behavior when the port receives a reset request.
19-18	SWAP_TX	R/W	VBUS_rst	2'b0	Software control for transmitter lane swap functionality for this port. This field initially indicates the sampled value of the SP_TX_SWAP pin. 0b00 = lanes 0,1,2,3 steered to 0,1,2,3 — no reversal 0b01 = lanes 0,1,2,3 steered to 1,0, 3,2 — 2x lane reversal 0b10 = lanes 0,1,2,3 steered to 3,2,1,0 — 4x lane reversal 0b11 = lanes 0,1,2,3 steered to 2,3,0,1 — combination reversal This field only controls the lanes associated with this register's port: • The lanes of 1x ports are unaffected by this register • The lanes of 2x ports are only reversed by the 2x lane reversal and only the two lanes of the port are reversed • The lanes of 4x ports are affected by all settings of this register. For more information, see Lane Reversal. Note: This field must not be changed after boot_complete is asserted.

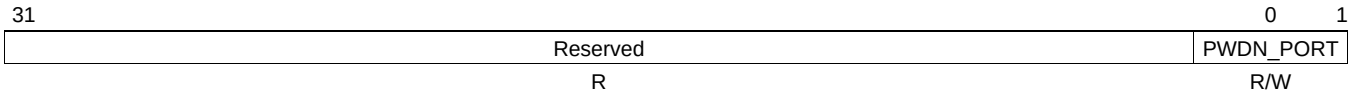
Table 3-110. PLM Port(n) Implementation Specific Control Register (Address offset 0x1B080, 0x1B100, 0x1B180, 0x1B200) (continued)

Bit	Name	R/W	Reset Source	Reset Value	Description
17-16	SWAP_RX	R/W	VBUS_rst	2'b0	Software control for receiver lane swap functionality for this port. This field initially indicates the sampled value of the SP_RX_SWAP pin. 0b00 = lanes 0,1,2,3 steered to 0,1,2,3 — no reversal 0b01 = lanes 0,1,2,3 steered to 1,0, 3,2 — 2x lane reversal 0b10 = lanes 0,1,2,3 steered to 3,2,1,0 — 4x lane reversal 0b11 = lanes 0,1,2,3 steered to 2,3,0,1 — combination reversal This field only controls the lanes associated with this register's port: • The lanes of 1x ports are unaffected by this register • The lanes of 2x ports are only reversed by the 2x lane reversal and only the two lanes of the port are reversed • The lanes of 4x ports are affected by all settings of this register. For more information, see Lane Reversal. Note: This field must not be changed after boot complete is asserted.
15-0	DLT_THRESH	R/W	VBUS_rst	0b0	Sets the threshold for the dead link timer.

3.18.3 PLM Port Powerdown Control Register (PLM_SP(n)_PWDN_CTL)

This port register provides implementation-specific control implementation of the port.

Figure 3-117. PLM Powerdown Control Register—PLM_SP(n)_PWDN_CTL



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-111. PLM Powerdown Control Register Field Description

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		31'b0	
0	PWDN_PORT	R/W	VBUS_rst	1'b0	Power down this port. Powering down a port • 0 = Normal mode of operation • 1 = The port is powered down Powering down the port takes these actions: • 1. The ip_clkN for the port is gated • 2. Logic and configuration registers associated with the port are • reset • 3. Sticky registers are optionally reset (see RapidIO Register Reset Control CSR.CLEAR_STICKY) The reset value of this field is set by srio_portN_power_down_in. The value of this field is provided on srio_portN_power_down_out.

3.18.4 PLM Port(n) Event Status Register(PLM_SP(n)_Status)

This register reports events detected by the PLM that are not reported in registers defined by the *RapidIO Interconnect Specification* (Revision 2.1).

Figure 3-118. PLM Port(n) Status Register (Address offset 0x1B090, 0x1B110, 0x1B190, 0x1B210)—PLM_SP(n)_STATUS

31	30	29	28	27	26	25	24	23	17
MAX_DENIAL	Reserved		LINK_INIT	DLT	PORT_ERR	OUTPUT_FAIL	OUTPUT_DEGR	Rsvd	
R/W	R		R/W	R/W	R/W	R/W	R/W	R	

16	15	14	13	12	11	10	9	0
RST_REQ	PBM_PW	TLM_PW	Reserved	MECS	PBM_INT	TLM_INT	Reserved	
R/W	R	R	R	R/W	R	R	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-112. PLM Port(n) Status Register (Address offset 0x1B090, 0x1B110, 0x1B190, 0x1B210)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	MAX_DENIAL	R/W1CS	VBUS_rst	0b0	Maximum denial error
30-29	Reserved	R		0b0	Reserved
28	LINK_INIT	R/W1CS	VBUS_rst	0b0	Link initialization notification
27	DLT	R/W1CS	VBUS_rst	0b0	Dead link timer event
26	PORT_ERR	R/W1CS	VBUS_rst	0b0	This bit indicates the status and operation of the RapidIO Port {0..3} Error and Status CSR.PORT_ERR bit.
25	OUTPUT_FAIL	R/W1CS	VBUS_rst	0b0	This bit indicates the status and operation of the RapidIO Port {0..3} Error and Status CSR.OUTPUT_FAIL bit.
24	OUTPUT_DEGR	R/W1CS	VBUS_rst	0b0	This bit indicates the status and operation of the RapidIO Port {0..3} Error and Status CSR.OUTPUT_DEGR bit.
23-17	Reserved	R		0b0	Reserved
16	RST_REQ	R/W1CS	VBUS_rst	0b0	Inbound reset request received
15	PBM_PW	R	VBUS_rst	0b0	Physical buffer module has detected an event for this port which uses port-write notification
14	TLM_PW	R	VBUS_rst	0b0	Transport layer module has detected an event for this port which uses port-write notification
13	Reserved	R		0b0	Reserved
12	MECS	R/W1CS	VBUS_rst	0b0	The port received a multicast-event control symbol
11	PBM_INT	R	VBUS_rst	0b0	Physical buffer module has detected an event for this port which uses interrupt notification
10	TLM_INT	R	VBUS_rst	0b0	Transport layer module has detected an event for this port which uses interrupt notification
9-0	Reserved	R		0b0	Reserved

3.18.5 PLM Port(n) Interrupt Enable Register (PLM_SP(n)_INT_ENABLE)

This register enables events reported in the RapidIO PLM Port {0..3} Event Status Register to contribute to the port's interrupt request.

Figure 3-119. PLM Port(n) Interrupt Enable Register (Address offset 0x1B094, 0x1B114, 0x1B194, 0x1B214) —PLM_SP(n)_INT_ENABLE

31	30 29	28	27	26	25	24	23 0
MAX_DENIAL	Rsvd	LINK_INIT	DLT	PORT_ERR	OUTPUT_FAIL	OUTPUT_DEGR	Rsvd
R/W	R	R/W	R/W	R/W	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-113. PLM Port(n) Interrupt Enable Register (Address offset 0x1B094, 0x1B114, 0x1B194, 0x1B214)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	MAX_DENIAL	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for MAX_DENIAL events.
30-29	Reserved	R		0b0	Reserved
28	LINK_INIT	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for LINK_INIT events.
27	DLT	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for DLT events.
26	PORT_ERR	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for PORT_ERR events.
25	OUTPUT_FAIL	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for OUTPUT_FAIL events.
24	OUTPUT_DEGR	R/W	VBUS_rst	0b0	When set, this bit enables interrupt notification for OUTPUT_DEGR events.
23-0	Reserved	R		0b0	Reserved

3.18.6 PLM Port(n) Port-Write Enable Register (PLM_SP(n)_PW_ENABLE)

This register enables events reported in the RapidIO PLM Port {0..3} Event Status Register to contribute to the port's port-write request.

Figure 3-120. PLM Port(n) Port-Write Enable Register (Address offset 0x1B098, 0x1B118, 0x1B198, 0x1B218) —PLM_SP(n)_PW_ENABLE

31	30	29	28	27	26	25	24	23 0
MAX_DENIAL	Rsvd		LINK_INIT	DLT	PORT_ERR	OUTPUT_FAIL	OUTPUT_DEGR	Rsvd
R/W	R		R/W	R/W	R/W	R/W	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-114. PLM Port(n) Port-Write Enable Register (Address offset 0x1B098, 0x1B118, 0x1B198, 0x1B218)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	MAX_DENIAL	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for MAX_DENIAL events.
30-29	Reserved	R		0b0	Reserved
28	LINK_INIT	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for LINK_INIT events.
27	DLT	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for DLT events.
26	PORT_ERR	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for PORT_ERR events.
25	OUTPUT_FAIL	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for OUTPUT_FAIL events.
24	OUTPUT_DEGR	R/W	VBUS_rst	0b0	When set, this bit enables Port-Write notification for OUTPUT_DEGR events.
23-0	Reserved	R		0b0	Reserved

3.18.7 PLM Port(n) Event Generate Register (PLM_SP(n)_EVENT_GEN)

This register generates PLM events for software verification.

Figure 3-121. PLM Port(n) Event Generate Register (Address offset 0x1B09C, 0x1B11C, 0x1B19C, 0x1B21C) —PLM_SP(n)_EVENT_GEN

31	30	29	28	27	26	25	24
MAX_DENIAL	Reserved		LINK_INIT	DLT	PORT_ERR	OUTPUT_FAIL	OUTPUT_DEGR
R/W	R		R/W	R/W	R/W	R/W	R/W

23	17	16	15	0
Reserved		RST_REQ	Reserved	
R		R/W	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-115. PLM Port(n) Event Generate Register (Address offset 0x1B09C, 0x1B11C, 0x1B19C, 0x1B21C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	MAX_DENIAL	R/W	VBUS_rst	0b0	Writing 1 to this bit creates a MAX_DENIAL event.
30-29	Reserved	R		0b0	Reserved
28	LINK_INIT	R/W	VBUS_rst	0b0	Writing 1 to this bit creates a LINK_INIT event.
27	DLT	R/W	VBUS_rst	0b0	Writing 1 to this bit creates a DLT event.
26	PORT_ERR	R/W	VBUS_rst	0b0	Writing 1 to this bit creates an PORT_ERR event. Note that this sets the RapidIO Port(n) Error and Status CSR.PORT_ERR bit. For the behavior of the port when PORT_ERR is asserted, see the definition in RapidIO Port(n) Error and Status CSR.
25	OUTPUT_FAIL	R/W	VBUS_rst	0b0	Writing 1 to this bit creates an OUTPUT_FAIL event. Note that this sets the RapidIO Port(n) Error and Status CSR.OUTPUT_FAIL bit. For the behavior of the port when OUTPUT_FAIL is asserted, see the definition in RapidIO Port(n) Error and Status CSR.
24	OUTPUT_DEGR	R/W	VBUS_rst	0b0	Writing 1 to this bit creates a OUTPUT_DEGR event.
23-17	Reserved	R		0b0	Reserved
16	RST_REQ	R/W1s	VBUS_rst	0b0	Writing 1 to this bit creates a RST_REQ event. Setting this event is equivalent to receiving a reset request from this port. The reset request is handled as programmed in the RapidIO PLM Port(n) Implementation-Specific Control Register SELF_RST and PORT_SELF_RST bits.
15-0	Reserved	R		0b0	Reserved

3.18.8 PLM Port(n) All Interrupts Enable Register (PLM_SP(n)_ALL_INT_EN)

This register provides interrupt enable control for the port.

Figure 3-122. PLM Port(n) All interrupts Enable Register (Address offset 0x1B0A0, 0x1B120, 0x1B1A0, 0x1B220)—PLM_SP(n)_ALL_INT_EN

31	1	0
Reserved		IRQ_EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-116. PLM Port(n) All interrupts Enable Register (Address offset 0x1B0A0, 0x1B120, 0x1B1A0, 0x1B220)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		0b0	Reserved
0	IRQ_EN	R/W	VBUS_rst	0b0	Interrupt error reporting enable

3.18.9 PLM Port(n) All Port-Writes Enable Register (PLM_SP(n)_ALL_PW_EN)

This register provides port-write enable control for the port.

Figure 3-123. PLM Port(n) All Port-Write Enable Register (Address offset 0x1B0A4, 0x1B124, 0x1B1A4, 0x1B224)—PLM_SP(n)_ALL_PW_EN

31	1	0
Reserved		PW_EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-117. PLM Port(n) All Port-Write Enable Register (Address offset 0x1B0A4, 0x1B124, 0x1B1A4, 0x1B224)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		0b0	Reserved
0	PW_EN	R/W	VBUS_rst	0b1	Interrupt error reporting enable

3.18.10 PLM Port(n) Path Control Register (PLM_SP(n)_PATH_CTL)

This implementation-specific per-path register indicates the number of channel resources, and controls the allocation of those resources to ports.

Figure 3-124. PLM Port(n) Path Control Register (Address offset 0x1B0B0, 0x1B130, 0x1B1B0, 0x1B230) —PLM_SP(n)_PATH_CTL

31 21	20 16	15 11	10 8	7 3	2 0
Reserved	PATH_ID	Reserved	PATH_CONFIG	Reserved	PATH_MODE
R	R	R	R	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-118. PLM Port(n) Path Control Register (Address offset 0x1B0b0, 0x1B130, 0x1B1b0, 0x1B230)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-21	Reserved	R		0b0	Reserved
20-16	PATH_ID	R	VBUS_rst	0b0	Identifies the path in which this port resides
15-11	Reserved	R		0b0	Reserved
10-8	PATH_CONFIG	R	VBUS_rst	3'b100	Indicates the path's configuration: • 01 = Configuration 1 - 1 lane, 1 port • 010 = Configuration 2 - 2 lanes, a maximum of 2 ports • 100 = Configuration 4 - 4 lanes, a maximum of 4 ports All other settings are reserved. See Figure 2-15 for more information.
7-3	Reserved	R		0b0	Reserved
2-0	PATH_MODE	R/W	VBUS_rst	3'b000	Sets the path's mode, which is constrained by • 000 = Mode 0 • 001 = Mode 1 • 010 = Mode 2 • 011 = Mode 3 • 100 = Mode 4 All other settings are reserved. The meaning of mode x is dependent on PATH_CONFIGURATION. Not all path modes are valid for all path configurations. If there is a conflict between the PATH_CONFIGURATION and the value written to this field, the field is left unchanged. The PATH_CONFIGURATION and PATH_MODEfields control the RapidIO Port {0..3} Control CSR.PORT_WIDTH field and limit the allowable values for the OVER_PWIDTH field for the ports implemented by the path. See Figure 2-15 for more information.

3.18.11 PLM Port(n) Discovery Timer Register (PLM_SP(n)_DISCOVERY_TIMER)

This per port (required for only port 0 and 2) register defines the discovery-timer value for RapidIO ports in 4x or 2x mode.

Figure 3-125. PLM Port(n) Discovery Timer Register (Address offset 0x1B0B4, 0x1B1B4)—PLM_SP(n)_DISCOVERY_TIMER

31 28	27 0
DISCOVERY_TIMER	Reserved
R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-119. PLM Port(n) Discovery Timer Register (Address offset 0x1B0B4, 0x1B1B4)

Bit	Name	R/W	Reset Value	Description
31-28	DISCOVERY_TIMER	R/W	0x7	If the RapidIO Port {0..3} Control CSR.PORT_WIDTH field indicates support for 4x or 2x mode, then this register exists for that port. This register does not exist for 1x ports. This field is used by RapidIO ports to configure the standard discovery timer value. In the *RapidIO Interconnect Specification* (Revision 2.1), the discovery timer is specified to be 28 msec +/- 4 msec (for Revision 1.3, this timer is 12 msec +/- 4 msec.). The timer allows time for the link partner to enter its discovery state, and if the link partner supports 4x mode, for all four lanes to be aligned. The discovery timer interval can be calculated with the following equation: Discovery timer interval [ms] = Prescaled IP_CLK * 52429 * DISCOVERY_TIMER where Prescaled IP_CLK period = IP_CLK period * RIO_IP_PRESCALAR_SRV_CLK value. For example, if IP_CLK = 156.25 Mhz, and if RIO_PRESCALAR_SRV_CLK = 16 and Discover_Timer = 5, this yields a Discovery Timer interval of: 1/156.25e6 * 16 * 52429 * 5 = 26.8 mS. Table 3-120 can assist you to select the Discovery Timer value. Note: Values of 0 and 1 are not legal.
0	Reserved	R/W	0b0	Reserved

Table 3-120. Discovery Timer Value

Default Timer Value (mS)	12	28				
Pre Divider	52429	52429				
IP_CLK Frequency (MHz)	IP_CLK Period (uS)	Prescale Period (uS)	Value for 12 mS Timer	Difference from 12 mS (mS)	Value for 28 mS Timer	Difference from 28 mS (mS)
312.5	0.003	0.0992	2	1.60	5	2.00
307.2	0.003	0.1009	2	1.42	5	1.55
250	0.004	0.1000	2	1.51	5	1.79
245.76	0.004	0.1017	2	1.33	5	1.33
156.25	0.006	0.1024	2	1.26	5	1.16
153.6	0.007	0.0977	2	1.76	5	2.40
125	0.008	0.1040	2	1.09	5	0.74
78.125	0.013	0.1024	2	1.26	5	1.16
76.8	0.013	0.1042	2	1.08	5	0.69
62.5	0.016	0.0960	2	1.93	6	2.20

3.18.12 Port (n) Silence Timer

This per port (required for only port 0 and 2) register defines the discovery-timer value for RapidIO ports in 4x or 2x mode.

Figure 3-126. Port n Silence Timer (Address offset 0x1B0B8, 0x1B138, 0x1B1B8, 0x1B238) —SP(n)_SILENCE_TIMER

31 28	27 0
SILENCE_TIMER	Reserved
R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-121. Port n Silence Timer (Address offset 0x1B0B8, 0x1B138, 0x1B1B8, 0x1B238)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-28	SILENCE_TIMER	R/W	VBUS_rst	0x1001	Silence Timer Defines the time that the port remains in the SILENT state. The default value of the timer depends upon the frequency of the reference clock for the port. Silence timer interval = Prescaled IP_CLK period x 410 x SILENCE_TIMER where the prescaled IP_CLK period = IP_CLK period * RIO_IP_PRESCALAR_SRV_CLK value. Note: a value of 0 is not legal.
0	Reserved	R		0b0	Reserved

3.18.13 PLM Port(n) Vmin Exponent Register (PLM_SP(n)_VMIN_EXP)

This per port register defines Vmin exponent.

Figure 3-127. PLM Port(n) Vmin Exponent Register (Address offset 0x1B0BC, 0x1B13C, 0x1B1BC, 0x1B23C) —PLM_SP(n)_VMIN_EXP

31 29	28 24	23 20	19 16	15 12	11 8	7 0
Reserved	VMIN_EXP	Reserved	IMAX	Reserved	MMAX	Reserved
R	R/W	R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-122. PLM Port(n) Vmin Exponent Register (Address offset 0x1B0bC, 0x1B13C, 0x1B1BC, 0x1B23C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-29	Reserved	R		0b0	Reserved
28-24	VMIN_EXP	R/W	VBUS_rst	0b0	VMIN_EXP sets the number of /VALID/ code-groups required for synchronization.
23-20	Reserved	R		0b0	Reserved
19-16	IMAX	R/W	VBUS_rst	0x3	IMAX sets the maximum threshold for the Icounter specified by *RapidIO Interconnect Specification* (Revision 2.1).
15-12	Reserved	R		0b0	Reserved
11-8	MMAX	R/W	VBUS_rst	0x3	MMAX sets the maximum threshold for the Mcounter specified by *RapidIO Interconnect Specification* (Revision 2.1).
7-0	Reserved	R		0b0	Reserved

3.18.14 PLM Port(n) Lane Polarity Control Register (PLM_SP(n)_POL_CTL)

This per-port register swaps the polarity of the differential pairs used by this port.

Figure 3-128. PLM Port(n) Lane Polarity Control Register (Address offset 0x1B0C0, 0x1B140, 0x1B1C0, 0x1B240) —PLM_SP(n)_POL_CTL

31					20	19	18
Reserved						TX3_POL	TX2_POL
17	16	15	4	3	2	1	0
TX1_POL	TX0_POL	Reserved		RX3_POL	RX2_POL	RX1_POL	RX0_POL

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-123. PLM Port(n) Lane Polarity Control Register (Address offset 0x1B0C0, 0x1B140, 0x1B1C0, 0x1B240)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-20	Reserved	R		0b0	Reserved
19	TX3_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
18	TX2_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
17	TX1_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
16	TX0_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
15-4	Reserved	R		0b0	Reserved
3	RX3_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
2	RX2_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
1	RX1_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.
0	RX0_POL	R/W	VBUS_rst	1'b0	Determines whether or not the lines that compose a differential pair are swapped.

3.18.15 PLM Port(n) Packet Denial Control Register (PLM_SP(n)_DENIAL_CTL)

This per-port register controls when a “too many retries” event is detected.

Figure 3-129. PLM Port(n) Denial Control Register (Address offset 0x1B0C8, 0x1B148, 0x1B1C8, 0x1B248) —PLM_SP(n)_DENIAL_CTL

31 30	29	28	27 8	7 0
Reserved	CNT_PNA	CNT_RTY	Reserved	DENIAL_THRESH
R	R/W	R/W	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-124. PLM Port(n) Denial Control Register (Address offset 0x1B0C8, 0x1B148, 0x1B1C8, 0x1B248)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-30	Reserved	R		0b0	Reserved
29	CNT_PNA	R/W	VBUS_rst	0b1	Controls whether packet-not-accepted control symbols count toward the packet denial threshold value
28	CNT_RTY	R/W	VBUS_rst	0b1	Controls whether retry control symbols count toward the packet denial threshold value
27-8	Reserved	R		0b0	Reserved
7-0	DENIAL_THRESH	R	VBUS_rst	0b0	Sets the threshold for reporting too many consecutive retries.

3.18.16 PLM Port(n) Received MECS Status Register (PLM_SP(n)_RCVD_MECS)

Captures the MECS CMD field values received by this RapidIO port. This register is provided for informational purposes only, and can only be cleared by software.

Figure 3-130. PLM Port(n) Received MECS Status Register (Address offset 0x1B0D0, 0x1B150, 0x1B1D0, 0x1B250) —PLM_SP(n)_RCVD_MECS

31	8	7	0
Reserved		CMD_STAT	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-125. PLM Port(n) Received MECS status Register (Address offset 0x1B0D0, 0x1B150, 0x1B1D0, 0x1B250)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	CMD_STAT	R/W1c	VBUS_rst	0b0	This bit field indicates which MECS commands have been received.

3.18.17 PLM Port(n) MECS Forwarding Register (PLM_SP(n)_MECS_FWD)

This register subscribes the port to forward MECS with CMD values which match those enabled in the SUBSCRIPTION field.

Figure 3-131. PLM Port(n) MECS Forwarding Register (Address offset 0x1B0D8, 0x1B158, 0x1B1D8, 0x1B258) —PLM_SP(n)_MECS_FWD

31	8	7	1	0
Reserved		SUBSCRIPTION		MULT_CS
R		R/W		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-126. PLM Port(n) MECS Forwarding Register (Address offset 0x1B0D8, 0x1B158, 0x1B1D8, 0x1B258)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-1	SUBSCRIPTION	R/W	VBUS_rst	0b0	Specifies which MECS CMD values the port should forward to its link.
0	MULT_CS	R/W	VBUS_rst	0b0	This bit is mapped to RapidIO Port {0..3} Control CSR.MULT_CS and performs the same function. Either offset can be used to access this bit.

3.18.18 PLM Port(n) Control Symbol Transmit 1 (PLM_SP(n)_LONG_CS_TX1)

Writing to this register controls the SType 0 and SType 1 values of the short or long control symbol to be sent. Writing to this register triggers transmission of a short control symbol when IDLE1 is in use.

Figure 3-132. PLM Port(n) Control Symbol Transmit 1 (Address offset 0x1B0E0, 0x1B160, 0x1B1E0, 0x1B260) —PLM_SP(n)_LONG_CS_TX1

31	30 28	27 26	25 20	19 18	17 12
Reserved	STYPE_0	Reserved	PAR_0	Reserved	PAR_1
R	R/W	R	R/W	R	R/W

11 9	8	7	6 4	3	2 0
Reserved	CS_EMB	Reserved	STYPE_1	Reserved	CMD
R	R/W	R	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-127. PLM Port(n) Control Symbol Transmit 1 (Address offset 0x1B0E0, 0x1B160, 0x1B1E0, 0x1B260)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	Reserved	R		0b0	Reserved
30-28	STYPE_0	R/W	VBUS_rst	0b0	Encoding for control symbol. This field uses the parameters PAR_0 and PAR_1.
27-26	Reserved	R		0b0	Reserved
25-20	PAR_0	R/W	VBUS_rst	0b0	Used in conjunction with SType0 encoding.
19-18	Reserved	R		0b0	Reserved
17-12	PAR_1	R/W	VBUS_rst	0b0	Used in conjunction with SType0 encoding.
11-9	Reserved	R		0b0	Reserved
8	CS_EMB	R/W	VBUS_rst	0b0	Embed the control symbol into a data stream
7	Reserved	R		0b0	Reserved
6-4	STYPE_1	R/W	VBUS_rst	0b0	Encoding for the control symbol that uses the CMD parameter.
3	Reserved	R		0b0	Reserved
2-0	CMD	R/W	VBUS_rst	0b0	Used in conjunction with SType1 encoding to define the link maintenance commands.

3.18.19 PLM Port(n) Control Symbol Transmit 2 (PLM_SP(n)_LONG_CS_TX2)

Writing to this register controls the SType2 bits of a long control symbol to be sent to the RapidIO link partner.

Figure 3-133. PLM Port(n) Control Symbol Transmit 2 (Address offset 0x1B0E4, 0x1B164, 0x1B1E4, 0x1B264) —PLM_SP(n)_LONG_CS_TX2

31	30 28	27	26 16	15 0
Reserved	STYPE2	Reserved	PARM	Reserved
R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-128. PLM Port(n) Control Symbol Transmit 2 (Address offset 0x1B0E4, 0x1B164, 0x1B1E4, 0x1B264)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	Reserved	R		0b0	Reserved
30-28	STYPE2	R/W	VBUS_rst	0b0	STYPE2 encoding, as defined in Part 12.
27	Reserved	R		0b0	Reserved
26-16	PARM	R/W	VBUS_rst	0b0	Used in conjunction with STYPE2 encoding.
15-0	Reserved	R		0b0	Reserved

3.18.20 Transport Layer Block Header Register (TLM_BH)

This register identifies the following block of registers as the IDT Transport Layer Device-Specific register block.

Figure 3-134. TLM Block Header (Address offset 0x1B300)—TLM_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-129. TLM Block Header (Address offset 0x1B300)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x0106	Pointer to the next IDT block, in units of 0x100 bytes.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.21 TLM Port(n) Control Register (TLM_SP(n)_CONTROL)

This register provides control of the TLM for a port.

Figure 3-135. TLM Port(n) Control Register (Address offset 0x1B380, 0x1B400, 0x1B480, 0x1B500)—TLM_SP(n)_CONTROL

31	30	29 28	27 22	21	20	19 16	15 12	11 0
Reserved	PORTGROUP_SELECT	VOQ _SELECT	Reserved	TGT_ID_DIS	MTC_TGT_ID _DIS	Reserved	LENGTH	Reserved
R	R/W	R/W	R	R/W	R/W	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-130. TLM Port(n) Control Register (Address offset 0x1B380, 0x1B400, 0x1B480, 0x1B500)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	Reserved	R		0b0	Reserved
30	PORTGROUP _SELECT	R/W	VBUS_rst	0b0	Configures the number of port groups to which the TLMi directs traffic which is not routed to the LLM.
29-28	VOQ_SELECT	R/W	VBUS_rst	0b0	On ingress, configures the number of VOQs within each port group.
27-22	Reserved	R		0b0	Reserved
21	TGT_ID_DIS	R/W	VBUS_rst	0b0	Used in conjunction with MTC_TGT_ID_DIS to configure promiscuous mode (see destID validation block at Table 3-131).
20	MTC_TGT_ID_DIS	R/W	VBUS_rst	0b0	Used in conjunction with TGT_ID_DIS to configure promiscuous mode (see destID validation block at Table 3-131).
19-16	Reserved	R		0b0	Reserved
15-12	LENGTH	R	VBUS_rst	0x9	Sets the number of data nodes that must be stored before PBMi notifies the scheduler of a packet to be scheduled.
11-0	Reserved	R		0b0	Reserved

Table 3-131. Behavior of TGT_ID_DIS and MTC_TGT_ID_DIS

Endpoint	TGT_ID_DIS	0	1	0	1
Maintenance Request/Reserved Packets	MTC_TGT_ID_DIS	0	0	1	1
	Matches base device ID	Route to LLM	Route to LLM	Route to LLM	Route to LLM
	Matches BRR	Route by BRR	Route by BRR	Route to LLM	Route to LLM
	No match	Discard	Route to user core	Route to LLM	Route to LLM
All packets other than Maintenance Request/Reserved Packets	Matches base device ID	Route to user core	Route to user core	Route to user core	Route to user core
	Matches BRR	Route to user core	Route to user core	Route to user core	Route to user core
	No match	Discard	Route to user core	Discard	Route to user core

3.18.22 TLM Port(n) Status Register (TLM_SP(n)_STATUS)

This implementation-specific per-path register indicates the number of channel resources, and controls the allocation of those resources to ports.

Figure 3-136. TLM Port(n) Status Register (Address offset 0x1B390, 0x1B410, 0x1B490, 0x1B510)—TLM_SP(n)_STATUS

31	30 21	20	19 0
IG_BAD_VC	Reserved	IG_BRR_FILTER	Reserved
R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-132. TLM Port(n) Status Register (Address offset 0x1B390, 0x1B410, 0x1B490, 0x1B510)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	IG_BAD_VC	R/W1CS	VBUS_rst	0b0	Detected an inbound packet with the VC bit set. The packet that sets this bit is not logged.
30-21	Reserved	R		0b0	Reserved
20	IG_BRR_FILTER	R/W1CS	VBUS_rst	0b0	Discarded an inbound transaction based on the BRR programming.
19-0	Reserved	R		0b0	Reserved

3.18.23 TLM Port(n) Interrupt Enable Register (TLM_SP(n)_INT_ENABLE)

This register enables per-port status of errors detected by TLM to notify the system host by interrupt.

Figure 3-137. TLM Port(n) Interrupt Enable Register (Address offset 0x1B394, 0x1B414, 0x1B494, 0x1B514) —TLM_SP(n)_INT_ENABLE

31	30 21	20	19 0
IG_BAD_VC	Reserved	IG_BRR_FILTER	Reserved
R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-133. TLM Port(n) Interrupt Enable Register (Address offset 0x1B394, 0x1B414, 0x1B494, 0x1B514)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	IG_BAD_VC	R/W	VBUS_rst	0b0	When set, enables IG_BAD_VC to contribute to the ports interrupt request.
30-21	Reserved	R		0b0	Reserved
20	IG_BRR_FILTER	R/W	VBUS_rst	0b0	When set, enables IG_BRR_FILTER to contribute to the ports interrupt request.
19-0	Reserved	R		0b0	Reserved

3.18.24 TLM Port(n) Port-Write Enable Register (TLM_SP(n)_PW_ENABLE)

This register enables per-port status of errors detected by TLM to notify the system host by port-write.

Figure 3-138. TLM Port(n) Port-Write Enable Register (Address offset 0x1B398, 0x1B418, 0x1B498, 0x1B518) —TLM_SP(n)_PW_ENABLE

31	30 21	20	19 0
IG_BAD_VC	Reserved	IG_BRR_FILTER	Reserved
R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-134. TLM Port(n) Port-Write Enable Register (Address offset 0x1B398, 0x1B418, 0x1B498, 0x1B518)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	IG_BAD_VC	R/W	VBUS_rst	0b0	When set, enables IG_BAD_VC to contribute to the port's port-write request.
30-21	Reserved	R		0b0	Reserved
20	IG_BRR_FILTER	R/W	VBUS_rst	0b0	When set, enables IG_BRR_FILTER to contribute to the port's port-write request.
19-0	Reserved	R		0b0	Reserved

3.18.25 TLM Port(n) Event Generate Register (TLM_SP(n)_EVENT_GEN)

This register generates TLM events for software verification.

Figure 3-139. TLM Port(n) Event Generate Register (Address offset 0x1B39C, 0x1B41C, 0x1B49C, 0x1B51C) —TLM_SP(n)_EVENT_GEN

31	30 21	20	19 0
IG_BAD_VC	Reserved	IG_BRR_FILTER	Reserved
R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-135. TLM Port(n) Event Generate Register (Address offset 0x1B39C, 0x1B41C, 0x1B49C, 0x1B51C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	IG_BAD_VC	R/W1s	VBUS_rst	0b0	Writing 1 to this bit sets the corresponding bit in the RapidIO TLM Port {0..3} Status register.
30-21	Reserved	R		0b0	Reserved
20	IG_BRR_FILTER	R/W1s	VBUS_rst	0b0	Writing 1 to this bit sets the corresponding bit in the RapidIO TLM Port {0..3} Status register.
19-0	Reserved	R		0b0	Reserved

3.18.26 TLM Port(n) Base Routing Register 0 Control Register (TLM_SP(n)_BRR_0_CTL)

This register controls how the Base Routing Register (BRR) is applied to inbound packets. This is BRR 0.

Figure 3-140. TLM Port(n) Base Routing Register 0 Control Register (Address offset 0x1B3A0, 0x1B420, 0x1B4A0, 0x1B520) —TLM_SP(n)_BRR_0_CTL

31	30 27	26	25	24	23 0
ENABLE	Reserved	ROUTE_MR_TO_LLM	Reserved	PRIVATE	Reserved
R/W	R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-136. TLM Port(n) Base Routing Register 0 Control Register (Address offset 0x1B3A0, 0x1B420, 0x1B4A0, 0x1B520)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	ENABLE	R/Ws	VBUS_rst	0b0	Enables the BRR for use in accepting and routing inbound packets.
30-27	Reserved	R		0b0	Reserved
26	ROUTE_MR_TO_LLM	R/Ws	VBUS_rst	0b1	When set, maintenance request/reserved packets with destIDs which match this BRR are routed to the LLM, otherwise they are routed to the user core.
25	Reserved	R		0b0	Reserved
24	PRIVATE	R/Ws	VBUS_rst	0b1	Configures the BRR to be used only by its port or by all ports in the path.
23-0	Reserved	R		0b0	Reserved

3.18.27 TLM Port(n) Base Routing Register 0 Pattern and Match Register (TLM_SP(n)_BRR_0_PATTERN_MATCH)

This register provides the pattern to which the inbound destID is compared, and which bits participate in the comparison.

Figure 3-141. TLM Port(n) Base Routing Register 0 Pattern and Match Register (Address offset 0x1B3A4, 0x1B424, 0x1B4A4, 0x1B524)

31	16	15	0
PATTERN		MATCH	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-137. TLM Port(n) Base Routing Register 0 Pattern and Match Register (Address offset 0x1B3A4, 0x1B424, 0x1B4A4, 0x1B524)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	PATTERN	R/Ws	VBUS_rst	0b0	PATTERN provides the 16 bits that are compared one-for-one against the inbound destID.
15-0	MATCH	R/Ws	VBUS_rst	0b0	MATCH indicates which of the 16 bits of the inbound destID are compared against PATTERN.

3.18.28 TLM Port(n) Base Routing Register 1 Control Register (TLM_SP(n)_BRR_1_CTL)

This register controls how the Base Routing Register (BRR) is applied to inbound packets. This is BRR 1.

Figure 3-142. TLM Port(n) Base Routing Register 1 Control Register (Address offset 0x1B3B0, 0x1B430, 0x1B4B0, 0x1B530) —TLM_SP(n)_BRR_1_CTL

31	30 27	26	25	24	23 0
ENABLE	Reserved	ROUTE_MR_TO_LLM	Reserved	PRIVATE	Reserved
R/W	R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-138. TLM Port(n) Base Routing Register 1 Control Register (Address offset 0x1B3B0, 0x1B430, 0x1B4B0, 0x1B530)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	ENABLE	R/Ws	VBUS_rst	0b0	Enables the BRR for use in accepting and routing inbound packets.
30-27	Reserved	R		0b0	Reserved
26	ROUTE_MR_TO_LLM	R/Ws	VBUS_rst	0b1	When set, maintenance request/reserved packets with destIDs which match this BRR are routed to the LLM, otherwise they are routed to the User Core.
25	Reserved	R		0b0	Reserved
24	PRIVATE	R/Ws	VBUS_rst	0b1	Configures the BRR to be used only by its port or by all ports in the path.
23-0	Reserved	R		0b0	Reserved

3.18.29 TLM Port(n) Base Routing Register 1 Pattern and Match Register (TLM_SP(n)_BRR_1_PATTERN_MATCH)

This register provides the pattern to which the inbound destID is compared, and which bits participate in the comparison.

Figure 3-143. TLM Port(n) Base Routing Register 1 Pattern and Match Register (Address offset 0x1B3B4, 0x1B434, 0x1B4B4, 0x1B534)

31	16	15	0
PATTERN		MATCH	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-139. TLM Port(n) Base Routing Register 1 Pattern and Match Register (Address offset 0x1B3B4, 0x1B434, 0x1B4B4, 0x1B534)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	PATTERN	R/Ws	VBUS_rst	0b0	PATTERN provides the 16 bits that are compared one-for-one against the inbound destID.
15-0	MATCH	R/Ws	VBUS_rst	0xFFFF	MATCH indicates which of the 16 bits of the inbound destID are compared against PATTERN.

3.18.30 TLM Port(n) Base Routing Register 2 Control Register (TLM_SP(n)_BRR_2_CTL)

This register controls how the Base Routing Register (BRR) is applied to inbound packets. This is BRR 2.

Figure 3-144. TLM Port(n) Base Routing Register 2 Control Register (Address offset 0x1B3C0, 0x1B440, 0x1B4C0, 0x1B540) —TLM_SP(n)_BRR_2_CTL

31	30 27	26	25	24	23 0
ENABLE	Reserved	ROUTE_MR_TO_LLM	Reserved	PRIVATE	Reserved
R/W	R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-140. TLM Port(n) Base Routing Register 2 Control Register (Address offset 0x1B3C0, 0x1B440, 0x1B4C0, 0x1B540)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	ENABLE	R/Ws	VBUS_rst	0b0	Enables the BRR for use in accepting and routing inbound packets.
30-27	Reserved	R		0b0	Reserved
26	ROUTE_MR_TO_LLM	R/Ws	VBUS_rst	0b1	When set, maintenance request/reserved packets with destIDs which match this BRR are routed to the LLM, otherwise they are routed to the User Core.
25	Reserved	R		0b0	Reserved
24	PRIVATE	R/Ws	VBUS_rst	0b1	Configures the BRR to be used only by its port or by all ports in the path.
23-0	Reserved	R		0b0	Reserved

3.18.31 TLM Port(n) Base Routing Register 2 Pattern and Match Register (TLM_SP(n)_BRR_2_PATTERN_MATCH)

This register provides the pattern to which the inbound destID is compared, and which bits participate in the comparison.

Figure 3-145. TLM Port(n) Base Routing Register 2 Pattern and Match Register (Address offset 0x1B3C4, 0x1B444, 0x1B4C4, 0x1B544)

31	16	15	0
PATTERN		MATCH	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-141. TLM Port(n) Base Routing Register 2 Pattern and Match Register (Address offset 0x1B3C4, 0x1B444, 0x1B4C4, 0x1B544)

Bit	Name	R/W	Reset source	Reset Value	Description
31-16	PATTERN	R/Ws	VBUS_rst	0b0	PATTERN provides the 16 bits that are compared one-for-one against the inbound destID.
15-0	MATCH	R/Ws	VBUS_rst	0xFFFF	MATCH indicates which of the 16 bits of the inbound destID are compared against PATTERN.

3.18.32 TLM Port(n) Base Routing Register 3 Control Register (TLM_SP(n)_BRR_3_CTL)

This register controls how the Base Routing Register (BRR) is applied to inbound packets. This is BRR 3.

Figure 3-146. TLM Port(n) Base Routing Register 3 Control Register (Address offset 0x1B3D0, 0x1B450, 0x1B4D0, 0x1B550) —TLM_SP(n)_BRR_3_CTL

31	30 27	26	25	24	23 0
ENABLE	Reserved	ROUTE_MR_TO_LLM	Reserved	PRIVATE	Reserved
R/W	R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-142. TLM Port(n) Base Routing Register 3 Control Register (Address offset 0x1B3D0, 0x1B450, 0x1B4D0, 0x1B550)

Bit	Name	R/W	Reset Source	Reset Value	Description
31	ENABLE	R/Ws	VBUS_rst	0b0	Enables the BRR for use in accepting and routing inbound packets.
30-27	Reserved	R		0b0	Reserved
26	ROUTE_MR_TO_LLM	R/Ws	VBUS_rst	0b1	When set, maintenance request/reserved packets with destIDs which match this BRR are routed to the LLM, otherwise they are routed to the user core.
25	Reserved	R		0b0	Reserved
24	PRIVATE	R/Ws	VBUS_rst	0b1	Configures the BRR to be used only by its port or by all ports in the path.
23-0	Reserved	R		0b0	Reserved

3.18.33 TLM Port(n) Base Routing Register 3 Pattern and Match Register (TLM_SP(n)_BRR_3_PATTERN_MATCH)

This register provides the pattern to which the inbound destID is compared, and which bits participate in the comparison.

Figure 3-147. TLM Port(n) Base Routing Register 3 Pattern and Match Register (Address offset 0x1B3D4, 0x1B454, 0x1B4D4, 0x1B554)

31	16	15	0
PATTERN		MATCH	
R/W		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-143. TLM Port(n) Base Routing Register 3 Pattern and Match Register (Address offset 0x1B3D4, 0x1B454, 0x1B4D4, 0x1B554)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	PATTERN	R/Ws	VBUS_rst	0b0	PATTERN provides the 16 bits that are compared one-for-one against the inbound destID.
15-0	MATCH	R/Ws	VBUS_rst	0xFFFF	MATCH indicates which of the 16 bits of the inbound destID are compared against PATTERN.

3.18.34 Packet Buffer Module Block Header Register (PBM_BH)

This register identifies the following block of registers as the IDT Packet Buffer Module-Specific register block.

Figure 3-148. PBM Block Header (Address offset 0x1B600)—PBM_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-144. PBM Block Header (Address offset 0x1B600)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x0109	Pointer to the next IDT block, in units of 0x100 bytes.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.35 PBM Port(n) Control Register (PBM_SP(n)_CONTROL)

This register must only be written before boot_complete is asserted, or while the port is in reset.

Figure 3-149. PBM Port(n) Control Register (Address offset 0x1B680, 0x1B700, 0x1B780, 0x1B800)—PBM_SP(n)_CONTROL

31 6	5 4	3	2 0
Reserved	EG_REORDER_MODE	Reserved	EG_REORDER_STICK
R	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-145. PBM Port(n) Control Register (Address offset 0x1B680, 0x1B700, 0x1B780, 0x1B800)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-6	Reserved	R		0b0	
5-4	EG_REORDER_MODE	R/W	VBUS_rst	2'b01	Selects the reorder action within PBMe
3	Reserved	R		0b0	Reserved
2-0	EG_REORDER_STICK	R/W	VBUS_rst	3'b0	Selects the number of repeat times the CRQ is reordered after each reorder event unless the CRQ has no unsent packets

3.18.36 PBM Port(n) Status Register (PBM_SP(n)_STATUS)

This register reports the status of various error conditions that are detected within the PBMi and PBMe.

Figure 3-150. PBM Port(n) Status Register (Address offset 0x1B690, 0x1B710, 0x1B790, 0x1B810)—PBM_SP(n)_STATUS

31	17	16	15
Reserved		IG_EMPTY	EG_EMPTY
R		R/W	R/W

14 5	4	3	2	1	0
Reserved	EG_DATA_OVERFLOW	EG_CRQ_OVERFLOW	Rsvd	EG_BAD_CHANNEL	EG_BABBLE_PACKET
R	R/W	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-146. PBM Port(n) Status Register (Address offset 0x1B690, 0x1B710, 0x1B790, 0x1B810)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-17	Reserved	R		0b0	Reserved
16	IG_EMPTY	R	VBUS_rst	0b1	The PBMi has no packets enqueued.
15	EG_EMPTY	R	VBUS_rst	0b1	The PBMe has not packets enqueued.
14-5	Reserved	R		0b0	Reserved
4	EG_DATA_OVERFLOW	R/W1cs	VBUS_rst	0b0	PBMe received a request to enqueue a packet for which it did not have enough data storage.
3	EG_CRQ_OVERFLOW	R/W1cs	VBUS_rst	0b0	PBMe received a request to enqueue a packet for which it did not have a CRQ entry.
2	Reserved	R		0b0	Reserved
1	EG_BAD_CHANNEL	R/W1cs	VBUS_rst	0b0	PBMe received a request to enqueue a packet on a channel enqueue interface which should be unused for the path's mode.
0	EG_BABBLE_PACKET	R/W1cs		0b0	PBMe detected a packet that exceeded 276 bytes on its enqueue interface.

3.18.37 PBM Port(n) Interrupt Enable Register (PBM_SP(n)_INT_ENABLE)

This register enables per port status of errors detected by PBM to notify the system host by interrupt.

Figure 3-151. PBM Port(n) Interrupt Enable Register (Address offset 0x1B694, 0x1B714, 0x1B794, 0x1B814)—PBM_SP(n)_INT_ENABLE

31 5	4	3	2	1	0
Reserved	EG_DATA_OVERFLOW	EG_CRQ_OVERFLOW	Rsvd	EG_BAD_CHANNEL	EG_BABBLE_PACKET
R	R/W	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-147. PBM Port(n) Interrupt Enable Register (Address offset 0x1B694, 0x1B714, 0x1B794, 0x1B814)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-5	Reserved	R		0b0	Reserved
4	EG_DATA_OVERFLOW	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's interrupt request.
3	EG_CRQ_OVERFLOW	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's interrupt request.
2	Reserved	R		0b0	Reserved
1	EG_BAD_CHANNEL	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's interrupt request.
0	EG_BABBLE_PACKET	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's interrupt request.

3.18.38 PBM Port(n) Port-Write Enable Register (PBM_SP(n)_Port-Write_ENABLE)

This register enables per-port status of errors detected by PBM to notify the system host by port-write.

Figure 3-152. PBM Port(n) Port-Write Enable Register (Address offset 0x1B698, 0x1B718, 0x1B798, 0x1B818) —PBM_SP(n)_Port-Write_ENABLE

31 5	4	3	2	1	0
Reserved	EG_DATA_OVERFLOW	EG_CRQ_OVERFLOW	Rsvd	EG_BAD_CHANNEL	EG_BABBLE_PACKET
R	R/W	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-148. PBM Port(n) Port-Write Enable Register (Address offset 0x1B698, 0x1B718, 0x1B798, 0x1B818)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-5	Reserved	R		0b0	Reserved
4	EG_DATA_OVERFLOW	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's port-write request.
3	EG_CRQ_OVERFLOW	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's port-write request.
2	Reserved	R		0b0	Reserved
1	EG_BAD_CHANNEL	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's port-write request.
0	EG_BABBLE_PACKET	R/W	VBUS_rst	0b0	When set, enables this event to contribute to the port's port-write request.

3.18.39 PBM Port(n) Event Generate Register (PBM_SP(n)_EVENT_GEN)

This register generates PBM events for software verification.

Figure 3-153. PBM Port(n) Event Generate Register (Address offset 0x1B69C, 0x1B71C, 0x1B79C, 0x1B81C) —PBM_SP(n)_EVENT_GEN

31 5	4	3	2	1	0
Reserved	EG_DATA_OVERFLOW	EG_CRQ_OVERFLOW	Rsvd	EG_BAD_CHANNEL	EG_BABBLE_PACKET
R	R/W	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-149. PBM Port(n) Event Generate Register (Address offset 0x1B69C, 0x1B71C, 0x1B79C, 0x1B81C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-5	Reserved	R		0b0	Reserved
4	EG_DATA_OVERFLOW	R/W1s	VBUS_rst	0b0	Setting this bit causes the corresponding bit in the RapidIO PBM Port {0..3} Status register to be set.
3	EG_CRQ_OVERFLOW	R/W1s	VBUS_rst	0b0	Setting this bit causes the corresponding bit in the RapidIO PBM Port {0..3} Status register to be set.
2	Reserved	R		0b0	Reserved
1	EG_BAD_CHANNEL	R/W1s	VBUS_rst	0b0	Setting this bit causes the corresponding bit in the RapidIO PBM Port {0..3} Status register to be set.
0	EG_BABBLE_PACKET	R/W1s	VBUS_rst	0b0	Setting this bit causes the corresponding bit in the RapidIO PBM Port {0..3} Status register to be set.

3.18.40 PBM Port(n) Ingress Resources Register (PBM_SP(n)_IG_RESOURCES)

This register indicates the number of data nodes and tags available for the port. The amount of resources is determined by compile-time parameters.

Figure 3-154. PBM Port(n) Ingress Resources Register (Address offset 0x1B6A0, 0x1B720, 0x1B7A0, 0x1B820) —PBM_SP(n)_IG_RESOURCES

31 26	25 16	15 10	9 0
Reserved	DATANODES	Reserved	TAGS
R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-150. PBM Port(n) Ingress Resources Register (Address offset 0x1B6A0, 0x1B720, 0x1B7A0, 0x1B820)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	DATANODES	R	VBUS_rst	10'h48	Indicates the number of data nodes implemented in the PBMi for packet storage for the port.
15-10	Reserved	R		0b0	Reserved
9-0	TAGS	R	VBUS_rst	10'h48	Indicates the number of tags implemented in the PBMi for packet storage for the port.

3.18.41 PBM Port(n) Egress Resources Register (PBM_SP(n)_EG_RESOURCES)

This register indicates the number of data nodes and tags available for the port.

Figure 3-155. P) —) —BM Port(n) Egress Resources Register (Address offset 0x1B6A4, 0x1B724, 0x1B7A4, 0x1B824) —PBM_SP(n)_EG_RESOURCES

31 26	25 16	15 7	6 0
Reserved	DATANODES	Reserved	CRQ_ENTRIES
R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-151. PBM Port(n) Egress Resources Register (Address offset 0x1B6A4, 0x1B724, 0x1B7A4, 0x1B824)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	DATANODES	R	VBUS_rst	10'h49	Indicates the number of data nodes implemented in the PBMe for packet storage for the port.
15-7	Reserved	R		0b0	Reserved
6-0	CRQ_ENTRIES	R	VBUS_rst	Port0 = 7'h21 Port1 = 7'h11 Port2 = 7'h09 Port3 = 7'h09	Indicates the number of CRQ entries implemented in the PBMe for packet storage for the port.

3.18.42 PBM Port(n) Ingress Watermarks 0 Register (PBM_SP(n)_IG_WATERMARK0)

This register allocates data nodes and tags for priority 1 packets. This register must only be changed while boot_complete is deasserted, or while the port is held in reset.

Figure 3-156. PBM Port(n) Ingress Watermarks 0 Register (Address offset 0x1B6B0, 0x1B730, 0x1B7B0, 0x1B830) —PBM_SP(n)_IG_WATERMARK0

31 26	25 16	15 10	9 0
Reserved	PRIO0CRF_WM	Reserved	PRIO0_WM
R	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-152. PBM Port(n) Ingress Watermarks 0 Register (Address offset 0x1B6B0, 0x1B730, 0x1B7B0, 0x1B830)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	PRIO0CRF_WM	R/W	VBUS_rst	0x3F	Priority 0+CRF packets are accepted if the number of free data nodes is greater than or equal to this value.
15-10	Reserved	R		0b0	Reserved
9-0	PRIO0_WM	R/W	VBUS_rst	0x048	Priority 0 packets are accepted if the number of free data nodes is greater than or equal to this value.

3.18.43 PBM Port(n) Ingress Watermarks 1 Register (PBM_SP(n)_IG_WATERMARK1)

This register allocates data nodes and tags for priority 1 packets. This register must only be changed while boot_complete is deasserted, or while the port is held in reset.

Figure 3-157. PBM Port(n) Ingress Watermarks 1 Register (Address offset 0x1B6B4, 0x1B734, 0x1B7B4, 0x1B834) —PBM_SP(n)_IG_WATERMARK1

31 16	25 16	15 10	9 0
Reserved	PRIO1CRF_WM	Reserved	PRIO1_WM
R	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-153. PBM Port(n) Ingress Watermarks 1 Register (Address offset 0x1B6B4, 0x1B734, 0x1B7B4, 0x1B834)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	PRIO1CRF_WM	R/W	VBUS_rst	0x02D	Priority 1+CRF packets are accepted if the number of free data nodes is greater than or equal to this value.
15-10	Reserved	R		0b0	Reserved
9-0	PRIO1_WM	R/W	VBUS_rst	0x036	Priority 1 packets are accepted if the number of free data nodes is greater than or equal to this value.

3.18.44 PBM Port(n) Ingress Watermarks 2 Register (PBM_SP(n)_IG_WATERMARK2)

This register allocates data nodes and tags for priority 2 packets. This register must only be changed while boot_complete is deasserted or while the port is held in reset.

Figure 3-158. PBM Port(n) Ingress Watermarks 2 Register (Address offset 0x1B6B8, 0x1B738, 0x1B7B8, 0x1B838) —PBM_SP(n)_IG_WATERMARK2

31 26	25 16	15 10	9 0
Reserved	PRIO2CRF_WM	Reserved	PRIO2_WM
R	R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

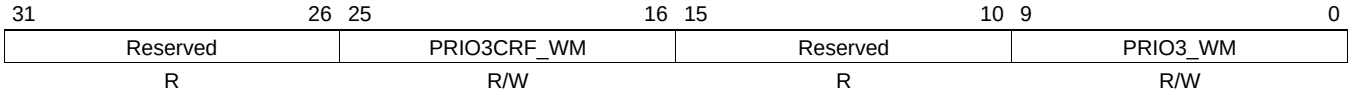
Table 3-154. PBM Port(n) Ingress Watermarks 2 Register (Address offset 0x1B6B8, 0x1B738, 0x1B7B8, 0x1B838)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	PRIO2CRF_WM	R/W	VBUS_rst	0x01B	Priority 2+CRF packets are accepted if the number of free data nodes is greater than or equal to this value.
15-10	Reserved	R		0b0	Reserved
9-0	PRIO2_WM	R/W	VBUS_rst	0x24	Priority 2 packets are accepted if the number of free data nodes is greater than or equal to this value.

3.18.45 PBM Port(n) Ingress Watermarks 3 Register (PBM_SP(n)_IG_WATERMARK3)

This register allocates data nodes and tags for priority 3 packets. This register must only be changed while boot_complete is deasserted, or while the port is held in reset.

Figure 3-159. PBM Port(n) Ingress Watermarks 3 Register (Address offset 0x1B6BC, 0x1B73C, 0x1B7BC, 0x1B83C) —PBM_SP(n)_IG_WATERMARK3



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-155. PBM Port(n) Ingress Watermarks 3 Register (Address offset 0x1B6BC, 0x1B73C, 0x1B7BC, 0x1B83C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-26	Reserved	R		0b0	Reserved
25-16	PRIO3CRF_WM	R/W	VBUS_rst	0x009	Priority 3+CRF packets are accepted if the number of free data nodes is greater than or equal to this value.
15-10	Reserved	R		0b0	Reserved
9-0	PRIO3_WM	R/W	VBUS_rst	0x12	Priority 3 packets are accepted if the number of free data nodes is greater than or equal to this value.

3.18.46 IDT-Specific Event Management Block Header (EM_BH)

This register identifies the following registers as the IDT RapidIO Event Management register block.

Figure 3-160. EM Block Header (Address offset 0x1B900)—EM_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

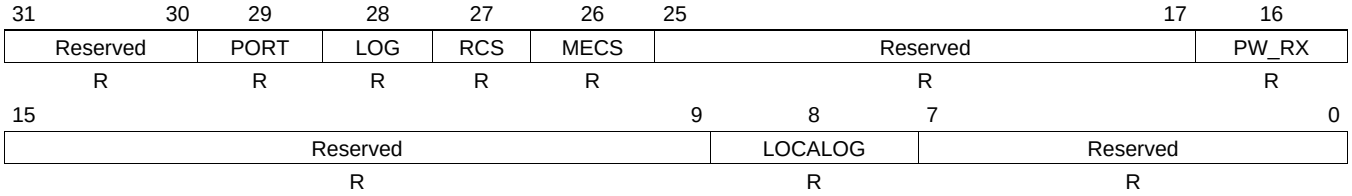
Table 3-156. EM Block Header (Address offset 0x1B900)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x010A	Pointer to the next IDT block, in units of 0x100 bytes. This points to the Port-Write Block Registers.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.47 Event Management Interrupt Status Register (EM_INT_STAT)

This register indicates the events, enabled lower in the hierarchy, that caused a RapidIO interrupt to be asserted (see event to interrupt/port-write hierarchy).

Figure 3-161. Event Management Interrupt Status Register (Address offset 0x1B910)—EM_INT_STAT



Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-157. Event Management Interrupt Status Register (Address offset 0x1B910)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-30	Reserved	R		0b0	Reserved
29	PORT	R	VBUS_rst	0b0	A port-specific event has been detected.
28	LOG	R	VBUS_rst	0b0	An enabled transport/logical layer error has been detected and reported by the user core through the user core logical layer error (LOGERR) signals.
27	RCS	R	VBUS_rst	0b0	A reset request has been detected by a port.
26	MECS	R	VBUS_rst	0b0	A MECS with an enabled CMD field (see RapidIO Event Management MECS Interrupt Enable Register) has been received by a RapidIO port.
25-17	Reserved	R		0b0	Reserved
16	PW_RX	R	VBUS_rst	0b0	A port-write has been received.
15-9	Reserved	R		0b0	Reserved
8	LOCALOG	R	VBUS_rst	0b0	Implementation-specific logical/transport layer error.
7-0	Reserved	R		0b0	Reserved

3.18.48 Event Management Interrupts Enable Register (EM_INT_ENABLE)

This register controls which events can assert a RapidIO interrupt.

NOTE: Interrupts must also be enabled in the RapidIO Event Management Device Interrupt Enable register.

Figure 3-162. Event Management Interrupts Enable Register (Address offset 0x1B914)—EM_INT_ENABLE

31 29	28	27	26	25 17	16	15 9	8	7 0
Reserved	LOG	Reserved	MECS	Reserved	PW_RX	Reserved	LOCALOG	Rsvd
R	R/W	R	R/W	R	R/W	R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-158. Event Management Interrupts Enable Register (Address offset 0x1B914)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-29	Reserved	R		0b0	Reserved
28	LOG	R/W	VBUS_rst	0b0	When set, this field enables a logical layer event detected in the user core to cause an interrupt.
27	Reserved	R		0b0	Reserved
26	MECS	R/W	VBUS_rst	0b0	When set, this field enables an interrupt to be raised upon reception of an MECS with a command value that is enabled in the RapidIO Event Management MECS Interrupt Enable register.
25-17	Reserved	R		0b0	Reserved
16	PW_RX	R/W	VBUS_rst	0b0	When set, this field enables the reception of a port-write to cause an interrupt.
15-9	Reserved	R		0b0	Reserved
8	LOCALOG	R/W	VBUS_rst	0b0	When set, this field enables an implementation-specific logical/transport layer error to cause an interrupt.
7-0	Reserved	R		0b0	Reserved

3.18.49 Event Management Interrupt Port Status Register (EM_INT_PORT_STAT)

This register defines which ports are asserting the interrupt notification method.

Figure 3-163. Event Management Interrupt Port Status Register (Address offset 0x1B918)—EM_INT_PORT_STAT

31	4	3	0
Reserved		IRQ_PENDING	
R		R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-159. Event Management Interrupt Port Status Register (Address offset 0x1B918)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-4	Reserved	R		0b0	Reserved
3-0	IRQ_PENDING	R	VBUS_rst	0b0	Per-port interrupt pending status.

3.18.50 Event Management Port-Write Status Register (EM_PW_STAT)

This register indicates the events, enabled lower in the hierarchy, that caused a RapidIO port-write to be sent (see event to interrupt/port-write hierarchy).

Figure 3-164. Event Management Port-Write Status Register (Address offset 0x1B920)—EM_PW_STAT

31 30	29	28	27	26 10	9	8	7 0
Reserved	PORT	LOG	RCS	Reserved	MULTIPORT_ERR	LOCALOG	Rsvd
R	R	R	R	R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-160. Event Management Port-Write Status Register (Address offset 0x1B920)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-30	Reserved	R		0b0	Reserved
29	PORT	R	VBUS_rst	0b0	A port-specific event has been detected.
28	LOG	R	VBUS_rst	0b0	An enabled transport/logical layer event has been detected and reported by the user core through the user core logical layer error (LOGERR) signals.
27	RCS	R	VBUS_rst	0b0	A reset request has been detected by a port.
26-10	Reserved	R		0b0	Reserved
9	MULTIPORT_ERR	R	VBUS_rst	0b0	Indicates that multiple ports have detected errors which use port-write notification.
8	LOCALOG	R	VBUS_rst	0b0	Implementation-specific local logical/transport layer error.
7-0	Reserved	R		0b0	Reserved

3.18.51 Event Management Port-Write Enable Register (EM_PW_ENABLE)

This register controls which events can cause a RapidIO port-write to be sent.

Figure 3-165. Event Management Port-Write Enable Register (Address offset 0x1B924)—EM_PW_ENABLE

31	29	28	27	9	8	7	0
Reserved		LOG	Reserved		LOCALOG	Reserved	
R		R/W	R		R/W	R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-161. Event Management Port-Write Enable Register (Address offset 0x1B924)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-29	Reserved	R		0b0	Reserved
28	LOG	R/W	VBUS_rst	0b1	When set, this field enables a logical layer event to cause a port-write to be sent.
27-9	Reserved	R		0b0	Reserved
8	LOCALOG	R/W	VBUS_rst	0b0	When set, this field enables an implementation-specific logical/transport layer error to cause a port-write to be sent.
7-0	Reserved	R		0b0	Reserved

3.18.52 Event Management Port-Write Port Status Register (EM_PW_PORT_STAT)

This register defines which ports are requesting port-writes to be sent. Refer to the ports' RapidIO PLM Port {0..3} Event Status register to determine the cause.

Figure 3-166. Event Management Port-Write Port Status Register (Address offset 0x1B928)—EM_PW_PORT_STAT

31	4	3	0
Reserved		PW_PENDING	
R		R	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-162. Event Management Port-Write Port Status Register (Address offset 0x1B928)

Name	Bit	R/W	Reset Source	Reset Value	Description
Reserved	31-4	R		0b0	Reserved
PW_PENDING	3-0	R	VBUS_rst	0b0	Per-port port-write pending status.

3.18.53 Event Management Device Interrupt Enable Register (EM_DEV_INT_EN)

This register controls whether each interrupt notification is enabled or disabled.

Figure 3-167. Event Management Device Interrupt Enable Register (Address offset 0x1B930)—EM_DEV_INT_EN

31	1	0
Reserved		INT_EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-163. Event Management Device Interrupt Enable Register (Address offset 0x1B930)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		0b0	Reserved
0	INT_EN	R/W	VBUS_rst	0b0	This bit controls whether the device interrupt line can be asserted.

3.18.54 Event Management Device Port-Write Enable Register (EM_DEV_PW_EN)

This register controls whether each port-write notification is enabled or disabled.

Figure 3-168. Event Management Device Port-Write Enable Register (Address offset 0x1B934)—EM_DEV_PW_EN

31	1	0
Reserved		PW_EN
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-164. Event Management Device Port-Write Enable Register (Address offset 0x1B934)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		0b0	Reserved
0	PW_EN	R/W	VBUS_rst	0b1	This bit controls whether any port-writes may be sent by this device.

3.18.55 Event Management MECS Status Register (EM_MECS_STAT)

Captures MECS CMD field values that have been received by SRIO-TEV2 from any RapidIO port, but not through mecs_trigger_in[7-0].

Figure 3-169. Event Management MECS Status Register (Address offset 0x1B93C)—EM_MECS_STAT

31	8	7	0
Reserved		CMD_STAT	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-165. Event Management MECS Status Register (Address offset 0x1B93C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	CMD_STAT	R/W1c	VBUS_rst	0b0	This bit field indicates which MECS commands were received by any RapidIO port.

3.18.56 Event Management MECS Interrupt Enable Register (EM_MECS_INT_EN)

This register enables a received MECS with corresponding CMD field value, as captured in RapidIO Event Management MECS Status register.

Figure 3-170. Event Management MECS Interrupt Enable Register (Address offset 0x1B940)—EM_MECS_INT_EN

31	8	7	0
Reserved		CMD_EN	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-166. Event Management MECS Interrupt Enable Register (Address offset 0x1B940)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	CMD_EN	R/W	VBUS_rst	0x01	Enables MECS with set CMD to raise an interrupt request

3.18.57 Event Management MECS Capture Out Register (EM_MECS_CAP_EN)

An edge is generated on the associated mecs_captured_out[7-0] pin when an MECS with a CMD field that is enabled in this register is received by any port, or is requested via the RapidIO Event Management MECS Request register.

Figure 3-171. Event Management MECS Capture Out Register (Address offset 0x1B944)—EM_MECS_CAP_EN

31	8	7	0
Reserved		CMD_EN	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-167. Event Management MECS Capture Out Register (Address offset 0x1B944)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	CMD_EN	R/W	VBUS_rst	0b0	Enables the associated bit of mecs_captured_out[7-0] to toggle.

3.18.58 Event Management MECS Trigger In Register (EM_MECS_TRIG_EN)

This register is associated with the mecs_trigger_in[7-0] top-level signal.

Figure 3-172. Event Management MECS Trigger In Register (Address offset 0x1B948)—EM_MECS_TRIG_EN

31 16	15 8	7 0
Reserved	CMD_STAT	CMD_EN
R	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-168. Event Management MECS Trigger In Register (Address offset 0x1B948)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	Reserved	R		0b0	Reserved
15-8	CMD_STAT	R/W1c	VBUS_rst	0b0	A bit is set whenever an edge is detected on corresponding mecs_trigger_in[7-0] pin
7-0	CMD_EN	R/W	VBUS_rst	0b0	Enables the associated bit of mecs_trigger_in[7-0] to trigger a MECS request to all RapidIO ports

3.18.59 Event Management MECS Request Register (EM_MECS_REQ)

This register enables one or many MECS of CMD values to be requested.

Figure 3-173. Event Management MECS Request Register (Address offset 0x1B94C)—EM_MECS_REQ

31	9	8	7	0
Reserved		SEND	CMD	
R		R/W	R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-169. Event Management MECS Request Register (Address offset 0x1B94C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-9	Reserved	R		0b0	Reserved
8	SEND	R/W1S	VBUS_rst	0b0	Writing 1 to this field initiates a MECS request for each CMD value set in the CMD field.
7-0	CMD	R/W	VBUS_rst	0b0	Setting this field indicates which MECS commands to be requested

3.18.60 Event Management MECS Port Status Register (EM_MECS_PORT_STAT)

This register flags any RapidIO port that has received a MECS (enabled or not). It is provided for informational purposes only, and can only be cleared directly by software.

Figure 3-174. Event Management MECS Port Status Register (Address offset 0x1B950)—EM_MECS_PORT_STAT

31	4	3	0
Reserved		PORT	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-170. Event Management MECS Port Status Register (Address offset 0x1B950)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-4	Reserved	R		0b0	Reserved
3-0	PORT	R/W1c	VBUS_rst	0b0	MECS Received status.

3.18.61 Event Management MECS Event Generate Register (EM_MECS_EVENT_GEN)

This register generates Event Management MECS events for software verification.

Figure 3-175. Event Management MECS Event Generate Register (Address offset 0x1B95C)—EM_MECS_EVENT_GEN

31	8	7	0
Reserved		CMD_STAT	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-171. Event Management MECS Event Generate Register (Address offset 0x1B95C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	CMD_STAT	R/W1S	VBUS_rst	0x000	Writing 1 to any bit sets the corresponding bit in the RapidIO Event Management MECS Status register.

3.18.62 Event Management Reset Request Port Status Register (EM_RST_PORT_STAT)

This register indicates on which ports a reset request was received, for ports configured to handle reset link-requests.

Figure 3-176. Event Management Reset Request Port Status Register (Address offset 0x1B960)—EM_RST_PORT_STAT

31	8	7	0
Reserved		RST_REQ	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-172. Event Management Reset Request Port Status Register (Address offset 0x1B960)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	RST_REQ	R/W1c	VBUS_rst	0b0	Per-port reset control symbol event received status.

3.18.63 Event Management Reset Request Interrupt Enable Register (EM_RST_INT_EN)

This register controls which ports cause an interrupt when they receive a reset request.

Figure 3-177. Event Management Reset Request Interrupt Enable Register (Address offset 0x1B968)—EM_RST_INT_EN

31	4	3	0
Reserved		RST_INT_EN	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-173. Event Management Reset Request Interrupt Enable Register (Address offset 0x1B968)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-4	Reserved	R		0b0	Reserved
3-0	RST_INT_EN	R/W	VBUS_rst	0b0	Per-port reset request event interrupt enable.

3.18.64 Event Management Reset Request Port-Write Enable Register (EM_RST_PW_EN)

This register controls which ports cause a port-write to be sent when they receive a reset request.

Figure 3-178. Event Management Reset Request Port-Write Enable Register (Address offset 0x1B970)—EM_RST_PW_EN

31	4	3	0
Reserved		RST_PW_EN	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-174. Event Management Reset Request Port-Write Enable Register (Address offset 0x1B970)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-4	Reserved	R		0b0	Reserved
3-0	RST_PW_EN	R/W	VBUS_rst	0b0	Per-port reset request port-write enable.

3.18.65 IDT-Specific Port-Write Block Header (PW_BH)

This register identifies the following registers as the IDT RapidIO port-write management register block.

Figure 3-179. PW Block Header (Address offset 0x1BA00)—PW_BH

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-175. PW Block Header (Address offset 0x1BA00)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x010D	Pointer to the next IDT block, in units of 0x100 bytes. This points to the LLM registers.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.66 Port-Write Control Register (PW_CTL)

This register controls aspects of port-write transmission and composition.

Figure 3-180. Port-Write Control Register (Address offset 0x1BA04)—PW_CTL

31 28	27 25	24	23 0
PW_TIMER	Reserved	PWC_MODE	Reserved
R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-176. Port-Write Control Register (Address offset 0x1BA04)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-28	PW_TIMER	R/W	VBUS_rst	0b0	Port-write timer
27-25	Reserved	R		0b0	Reserved
24	PWC_MODE	R/W	VBUS_rst	0b0	0b0 = Continuous port-write capture mode 0b1 = Reliable port-write capture mode
23-0	Reserved	R		0b0	Reserved

3.18.67 Port-Write Routing Register (PW_ROUTE)

This register defines on which port a port-write is transmitted.

Figure 3-181. Port-Write Routing Register (Address offset 0x1BA08)—PW_ROUTE

31	4	3	0
Reserved		PORT	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-177. Port-Write Routing Register (Address offset 0x1BA08)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-4	Reserved	R		0b0	Reserved
3-0	PORT	R/W	VBUS_rst	0b1	Indicates if a port-write should be sent to this port.

3.18.68 Port-Write Reception Status CSR (PW_RX_STAT)

This register indicates whether a port-write has been captured, and if a port-write had to be discarded due to lack of resources.

Figure 3-182. Port-Write Reception Status CSR (Address offset 0x1BA10)—PW_RX_STAT

31	16	15	12	11	9	8	7	4	3	2	1	0
Reserved		RW_SIZE		Rsvd		WDPTR	Rsvd		PW_SHORT	PW_TRUNC	PW_DISC	PW_VAL
R		R/W		R		R/W	R		R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-178. Port-Write Reception Status CSR (Address offset 0x1BA10)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	Reserved	R		0b0	Reserved
15-12	RW_SIZE	R/WS	VBUS_rst	0b0	WRSIZE, in combination with WPTR, determine the maximum size of the port-write received.
11-9	Reserved	R		0b0	Reserved
8	WDPTR	R/WS	VBUS_rst	0b0	WPTR, in combination with WRSIZE, determine the maximum size of the port-write received.
7-4	Reserved	R		0b0	Reserved
3	PW_SHORT	R/WS	VBUS_rst	0b0	Reserved
2	PW_TRUNC	R/WS	VBUS_rst	0b0	Reserved
1	PW_DISC	R/W1CS	VBUS_rst	0b0	A port-write was discarded. Clearing this bit has no other effect.
0	PW_VAL	R/W1CS	VBUS_rst	0b0	The port-write data registers contain valid data.

3.18.69 Port-Write Reception Event Generate Register (PW_RX_EVENT_GEN)

This register generates port-write reception events for software verification.

Figure 3-183. PW Reception Event Generate Register (Address offset 0x1BA14)—PW_RX_EVENT_GEN

31	2	1	0
Reserved		PW_DISC	PW_VAL
R		R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-179. PW Reception Event Generate Register (Address offset 0x1BA14)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-2	Reserved	R		0b0	Reserved
1	PW_DISC	R/W1S	VBUS_rst	0b0	Writing 1 to this sets the corresponding bit in the RapidIO Port-Write Reception Status CSR.
0	PW_VAL	R/W1S	VBUS_rst	0b0	Writing 1 to this sets the corresponding bit in the RapidIO Port-Write Reception Status CSR.

3.18.70 Port-Write Reception Capture(n) CSR (PW_RX_CAPT(n))

When a port-write maintenance request is received, the payload is captured in the RapidIO Port-Write Reception Capture 0-3 CSR registers, and an event is captured.

Figure 3-184. Port-Write Reception Capture(n) CSR (Address offset 0x1BA20, 0x1BA24, 0x1BA28, 0x1BA2C)—PW_RX_CAPT(n)

31	0
PW_CAPT	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-180. Port-Write Reception Capture(n) CSR (Address offset 0x1BA20, 0x1BA24, 0x1BA28, 0x1BA2C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-0	PW_CAPT	R/WS	VBUS_rst	0b0	Port-write payload, payload offset associated with register number.

3.18.71 IDT-Specific LLM Module Block Header (LLM_BH)

This register identifies the following registers as the IDT fabric module block.

Figure 3-185. LLM Module Block Header (Address offset 0x1BD00)—LLM_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-181. LLM Module Block Header (Address offset 0x1BD00)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x010E	Pointer to the next IDT block, in units of 0x100 bytes. This points to the Fabric Module registers.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.72 Whiteboard CSR (WHITEBOARD)

This register is provided for whiteboarding purposes. It does not impact the design in any way.

Figure 3-186. Whiteboard CSR (Address offset 0x1BD24)—WHITEBOARD

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-182. Whiteboard CSR (Address offset 0x1BD24)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-0	SCRATCH	R/W	VBUS_rst	0b0	The scratch field may be read and written with any value. The value written does no affect SRIO-TEV2's operation.

3.18.73 Port Number CSR (PORT_NUMBER)

NOTE: This register in NOT supported by TI.

This register is a copy of RapidIO Switch Port Information CAR, but is located on an 8-byte aligned address to allow a maintenance response, which returns its contents to be captured in the RapidIO Port(n) Packet/Control Symbol Error Capture CSR 0 and subsequent capture registers.

Figure 3-187. Port Number CSR (Address offset 0x1BD28)—PORT_NUMBER

31 16	15 8	7 0
Reserved	PORT_TOTAL	PORT_NUM
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-183. Port Number CSR (Address offset 0x1BD28)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	Reserved	R		0b0	Reserved
15-8	PORT_TOTAL	R	VBUS_rst	0b0	Port total
7-0	PORT_NUM	R	VBUS_rst	0b0	Port number

3.18.74 Port IP Prescalar for IP_CLK Register (RIO_PRESCALAR_SRV_CLK)

This register defines a prescalar for ip_clk to handle different clock frequencies.

Figure 3-188. Port IP Prescaler for IP_CLK Register (Address offset 0x1BD30)—RIO_IP_PRESCAL

31	8	7	0
Reserved		RIO_IP_PRESCALAR_SRV_CLK	
R		R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-184. Port IP Prescaler for IP_CLK Register (Address offset 0x1BD30)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-8	Reserved	R		0b0	Reserved
7-0	RIO_IP_PRESCALAR_SRV_CLK	R/W	VBUS_rst	0x1F	The default value of 31 is for an ip_clk frequency of 312.5 MHz. For different frequencies of ip_clk, use the formula: ip_clk_frequency (in MHz) /10 rounded to the nearest integer. Table 3-185 can help pick the value of the prescalar.

Table 3-185. RIO_IP_PRESCAL Values for Various ip_clk Frequencies

IP_CLK Frequency (MHz)	IP_CLK Period (uS)	RIO_IP_PRESCALAR_SRV_CLK	Prescaled IP_CLK period
312.5	0.003	31	0.0992
307.2	0.003	31	0.1009
250	0.004	25	0.1000
245.76	0.004	25	0.1017
156.25	0.006	16	0.1024
153.6	0.007	15	0.0977
125	0.008	13	0.1040
78.125	0.013	8	0.1024
76.8	0.013	8	0.1042
62.5	0.016	6	0.0960

3.18.75 Register Reset Control CSR (REG_RST_CTL)

This register configures register reset behavior.

Figure 3-189. Register Reset Control CSR (Address offset 0x1BD34)—REG_RST_CTL

31	1	0
Reserved		CLEAR_STICKY
R		R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-186. Register Reset Control CSR (Address offset 0x1BD34)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-1	Reserved	R		0b0	Reserved
0	CLEAR_STICKY	R/W	VBUS_rst	0b0	Allows the SELF_RST and PWDN_PORT resets to clear sticky register bits in addition to the normal configuration registers.

3.18.76 Local Logical/Transport Layer Error Detect CSR (LOCAL_ERR_DET)

This register is not to be confused with a similar register, RapidIO Logical/Transport Layer Error Detect CSR, which is implemented in the user core at the standard RapidIO offset. This register and the Local Logical/Transport Layer Error Capture registers are writable by software to allow software debug of the system error recovery methods.

Figure 3-190. Local Logical/Transport Layer Error Detect CSR (Address offset 0x1BD48)—LOCAL_ERR_DET

31 27	26	25 23	22	21 0
Reserved	ILL_ID	Rsvd	ILL_TYPE	Rsvd
R	R/W	R	R/W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-187. Local Logical/Transport Layer Error Detect CSR (Address offset 0x1BD48)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-27	Reserved	R		0b0	Reserved
26	ILL_ID	R/WS	VBUS_rst	0b0	Illegal transaction target error
25-23	Reserved	R		0b0	Reserved
22	ILL_TYPE	R/WS	VBUS_rst	0b0	Unsupported transaction
21-0	Reserved	R		0b0	Reserved

3.18.77 Local Logical/Transport Layer Error Enable CSR (LOCAL_ERR_EN)

This register contains the bits that control if an error condition locks the RapidIO Local Logical/Transport Layer Error Detect CSR and Capture registers (see registers starting at RapidIO Local Logical/Transport Layer High Address Capture CSR), and is reported to the system host through a port-write request or an interrupt.

Figure 3-191. Local Logical/Transport Layer Error Enable CSR (Address offset 0x1BD4C)—LOCAL_ERR_EN

31 27	26	25 23	22	21 0
Reserved	ILL_ID_EN	Reserved	ILL_TYPE_EN	Rsvd

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-188. Local Logical/Transport Layer Error Enable CSR (Address offset 0x1BD4C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-27	Reserved	R		0b0	Reserved
26	ILL_ID_EN	R/WS	VBUS_rst	0b0	Illegal transaction target error reporting enable
25-23	Reserved	R		0b0	Reserved
22	ILL_TYPE_EN	R/WS	VBUS_rst	0b0	Unsupported transaction reporting enable
21-0	Reserved	R		0b0	Reserved

3.18.78 Local Logical/Transport Layer High Address Capture CSR (LOCAL_H_ADDR_CAPT)

This register contains error information. It is locked when an local logical/transport error is detected and the corresponding enable bit is set. The contents of this register are only valid when the device is using 50- or 66-bit addressing.

Figure 3-192. Local Logical/Transport Layer High Address Capture CSR (Address offset 0x1BD50)—LOCAL_H_ADDR_CAPT

31	0
ADDR[31-0]	
R/W	

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-189. Local Logical/Transport Layer High Address Capture CSR (Address offset 0x1BD50)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-0	ADDR[31-0]	R/WS	VBUS_rst	0b0	Most significant bits of address associated with the error (for requests, not applicable for responses or FType 8 transactions). Address bits not used by the device are set to zero.

3.18.79 Local Logical/Transport Layer Address Capture CSR (LOCAL_ADDR_CAPT)

This register contains error information. It is locked when an local logical/transport error is detected and the corresponding enable bit is set.

Figure 3-193. Local Logical/Transport Layer Address Capture CSR (Address offset 0x1BD54)—LOCAL_ADDR_CAPT

31 3	2	1 0
ADDR[60-32]	Reserved	XAMSBS
R/W	R	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-190. Local Logical/Transport Layer Address Capture CSR (Address offset 0x1BD54)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-3	ADDR[60-32]	R/WS	VBUS_rst	0b0	Least significant 29 bits of address associated with the error.
2	Reserved	R		0b0	Reserved
1-0	XAMSBS	R/WS	VBUS_rst	0b0	Extended address bits of the address associated with the error (for requests, not applicable for responses or maintenance transactions)

3.18.80 Local Logical/Transport Layer deviceID Capture CSR (LOCAL_ID_CAPT)

This register contains error information. It is locked when an error is detected and the corresponding enable bit is set.

Figure 3-194. Local Logical/Transport Layer Deviceid Capture CSR (Address offset 0x1BD58)—LOCAL_ID_CAPT

31 24	23 16	15 8	7 0
MSB_DEST_ID	DEST_ID	MSB_SRC_ID	SRC_ID
R/W	R/W	R/W	R/W

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-191. Local Logical/Transport Layer deviceID Capture CSR (Address offset 0x1BD58)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-24	MSB_DEST_ID	R/WS	VBUS_rst	0b0	Most significant byte of destID associated with the error (Large transport system only)
23-16	DEST_ID	R/WS	VBUS_rst	0b0	destID associated with the error
15-8	MSB_SRC_ID	R/WS	VBUS_rst	0b0	Most significant byte of srcID associated with the error (Large transport system only)
7-0	SRC_ID	R/WS	VBUS_rst	0b0	srcID associated with the error

3.18.81 Local Logical/Transport Layer Control Capture CSR (LOCAL_CTRL_CAPT)

This register contains error information. It is locked when an error is detected and the corresponding enable bit is set.

Figure 3-195. Local Logical/Transport Layer Control Capture CSR (Address offset 0x1BD5C)—LOCAL_CTRL_CAPT

31 28	27 24	23 16	15 0
FTYPE	TTYPE	MESSAGE_INFO	Reserved
R?W	R?W	R?W	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-192. Local Logical/Transport Layer Control Capture CSR (Address offset 0x1BD5C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-28	FTYPE	R/WS	VBUS_rst	0b0	FType associated with the error
27-24	TTYPE	R/WS	VBUS_rst	0b0	TType associated with the error
23-16	MESSAGE_INFO	R/WS	VBUS_rst	0b0	Messages are not generated by SRIO-TEV2 and message information is not logged
15-0	Reserved	R		0b0	Reserved

3.18.82 IDT-Specific Fabric Module Block Header (FABRIC_BH)

This register identifies the following registers as the IDT fabric module block.

Figure 3-196. Fabric Module Block Header (Address offset 0x1BE00)—FABRIC_BH

31 16	15 12	11 0
NEXT_BLK_PTR	BLK_REV	BLK_TYPE
R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-193. Fabric Module Block Header (Address offset 0x1BE00)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-16	NEXT_BLK_PTR	R	VBUS_rst	0x0000	Pointer to the next IDT block, in units of 0x100 bytes. This points to NULL.
15-12	BLK_REV	R	VBUS_rst	0b0	IDT-defined revision of the block type in this device.
11-0	BLK_TYPE	R	VBUS_rst	0b0	IDT-defined identifier for the register block type.

3.18.83 Fabric Control and Status Register (FABRIC_CSR)

This register defines the Endpoint Fabric Module (EFM) Control and Status register. All control fields are shared among all ports. Status bits are used for debug purposes.

Figure 3-197. Fabric Control and Status Register (Address offset 0x1BE10)—FABRIC_CSR

31 28	27	26	25 0
Reserved	IG_LLM_BACKPRESSURE	IG_UC_BACKPRESSURE	Reserved
R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-194. Fabric Control and Status Register (Address offset 0x1BE10)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-28	Reserved	R		0b0	Reserved
27	IG_LLM_BACKPRESSURE	R	VBUS_rst	0b0	Ingress LLM backpressure indication
26	IG_UC_BACKPRESSURE	R	VBUS_rst	0b0	Ingress user core backpressure indication
25-0	Reserved	R		0b0	Reserved

3.18.84 Port(n) Fabric Status Register (SP(n)_FABRIC_STATUS)

This register defines the Port(n) Fabric Status Register for the endpoint fabric module (EFM). These status bits are used for debug purposes.

Figure 3-198. Port(n) Fabric Status Register (Address offset 0x1BE40, 0x1BE44, 0x1BE48, 0x1BE4C)—SP(n)_FABRIC_STATUS

31 24	23 20	19 16	15 0
Reserved	IG_PKT_ENABLE_STATUS	EG_PKT_ENABLE_STATUS	Reserved
R	R	R	R

Legend: R = Read only; W = Write only; - *n* = value after reset; -x, value is indeterminate — see the device-specific data manual

Table 3-195. Port(n) Fabric Status Register (Address offset 0x1BE40, 0x1BE44, 0x1BE48, 0x1BE4C)

Bit	Name	R/W	Reset Source	Reset Value	Description
31-24	Reserved	R		0b0	Reserved
23-20	IG_LLM_BACKPRESSURE	R	VBUS_rst	0b0	PBMi packet enable Indication
19-16	IG_UC_BACKPRESSURE	R	VBUS_rst	0b0	PBMe packet enable Indication
15-0	Reserved	R		0b0	Reserved

Examples

A.1 Channel Operations

```
#define SRIO_CFG_BASE                    (0x02900000u)
#define HOST_TX_COMPLETE_Q      (1000)
#define HOST_RX_FDQ              (2000)
#define SRIO_CDMA_TX_CHAN_REGION  (SRIO_CFG_BASE + 0x00001400u)
#define SRIO_CDMA_RX_CHAN_REGION  (SRIO_CFG_BASE + 0x00001800u)
#define CHANNEL0    (0)
#define ENABLE    (0)
#define DISABLE    (0x80000000)


/***** Transmit Channel Configuration Register Block *****/
#define CDMA_REG_TX_CHAN_CFG_A     0x000
#define CDMA_REG_TX_CHAN_CFG_B     0x004

/***** Tx Sched Config Register Block *****/
#define CDMA_REG_TX_SCHED_CHAN_CFG  0x000

/***** Enable Tx CDMA channel *****/
void enable_tx_chan (Uint32 base, Uint16 chan, Uint32 value)
{
  Uint32 *reg;
  reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_A + (chan * 32));
 *reg = value;
}


/***** Enable Rx CDMA channel *****/
void enable_rx_chan(Uint32 base, Uint16 chan, Uint32 value)
{
  Uint32 *reg;
  reg = (Uint32 *)(base + CDMA_REG_RX_CHAN_CFG_A + (chan * 32));
 *reg = value;
}

/***** Disable a TX CDMA channel & then Configure it *****/
void config_tx_chan (Uint32 base, Uint16 chan, Uint32 return_q)
{
  Uint32 *reg;

  reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_A + (chan * 32));
 *reg = 0; //disable the channel

  reg = (Uint32 *)(base + CDMA_REG_TX_CHAN_CFG_B + (chan * 32));
 *reg = return_q;
}


/***** Configure the priority of a TX CDMA channel *****/
void config_tx_sched (Uint32 base, Uint16 chan, Uint32 priority)
{
  Uint32 *reg;

  reg = (Uint32 *)(base + CDMA_REG_TX_SCHED_CHAN_CFG + (chan * 4));
```

```
 *reg = priority;
}


/***** Configure a RX CDMA channel flow *****/
void config_rx_flow(Uint32 base, Uint16 flow,
                    Uint32 a, Uint32 b, Uint32 c, Uint32 d,
                    Uint32 e, Uint32 f, Uint32 g, Uint32 h)
{
  Uint32 *reg;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_A + (flow * 32));
 *reg = a;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_B + (flow * 32));
 *reg = b;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_C + (flow * 32));
 *reg = c;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_D + (flow * 32));
 *reg = d;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_E + (flow * 32));
 *reg = e;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_F + (flow * 32));
 *reg = f;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_G + (flow * 32));
 *reg = g;

  reg = (Uint32 *)(base + CDMA_REG_RX_FLOW_CFG_H + (flow * 32));
 *reg = h;
}
```

A.2 Queue Operations

```
#define QMSS_CFG_BASE         (0x02a00000u)
#define QM_QMAN_REGION   (QMSS_CFG_BASE + 0x00020000u)
#define QM_STAT_REGION        (QMSS_CFG_BASE + 0x00000000u)

/***** Queue Management Region *****/
#define QM_REG_QUE_REG_A      0x000
#define QM_REG_QUE_REG_B      0x004
#define QM_REG_QUE_REG_C      0x008
#define QM_REG_QUE_REG_D      0x00c

/***** Queue Status Region *****/
#define QM_REG_QUE_STATUS_REG_A   0x000
#define QM_REG_QUE_STATUS_REG_B   0x004
#define QM_REG_QUE_STATUS_REG_C   0x008
#define QM_REG_QUE_STATUS_REG_D   0x00c

/************************************************
 * This function pushes descriptor info to a queue. *
 * Mode parameter:  1 = write register D only.      *
 *                  2 = write register C and D.     *
 ************************************************/
void push_queue(Uint16 qn, Uint8 mode, Uint32 c_val, Uint32 d_val)
{
  Uint32 *reg;

  if (mode == 2)
  {
```

```
    reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_C + (qn * 16));
   *reg = c_val;
  }

  reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_D + (qn * 16));
 *reg = d_val;
}


/***** This function pops a descriptor from a queue. *****/
Uint32 pop_queue (Uint16 qn)
{
  Uint32 *reg;
  Uint32  value;

  reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_D + (qn * 16));
  value = *reg;

  return (value);
}

/***** This function sets a queue threshold for triggering CDMA channels. *****/
void set_queue_threshold (Uint16 qn, Uint32 value)
{
  Uint32 *reg;

  reg = (Uint32 *)(QM_STAT_REGION + QM_REG_QUE_STATUS_REG_D + (qn * 16));
 *reg = value;
}


/***** This function returns the descriptor count of a queue *****/
Uint32 get_descriptor_count(Uint16 qn)
{
  Uint32 *reg;
  Uint32  count;

  reg = (Uint32 *)(QM_QMAN_REGION + QM_REG_QUE_REG_A + (qn * 16));
  count = *reg;

  return(count);
}
```

Software-Assisted Error Recovery

B.1 Error Recovery

After link initialization, the typical sequence is for the DSP software to check the *port_ok* bit in the SP(n)_ERR_STAT register before attempting to send packets. Without the *port_ok* bit being set, the two link partners are not initialized and cannot communicate. Additionally, before sending packets, check for error conditions and clear any error status bits in the SP(n)_ERR_STAT and SP(n)_ERR_DET registers. Most of these error indication bits are simply writable to clear, as described in their corresponding bit definitions. In some cases after reset, a given link partner may experience temporary bit errors resulting in OUTPUT ERROR-STOPPED or INPUT ERROR-STOPPED conditions. In these cases, hardware may or may not recover from this condition, depending on the states involved, severity, and location of the bit errors in the transmitted stream. Software can be used to immediately recover from the input and output error stopped states. TI recommends adding the following step after initialization and before trying to send any packets. This step can be added regardless of whether the device is currently in the output or input error-stopped states.

Software writes a value of 0x2003F044 into the PLM Port n Control Symbol Transmit 1 register (RIO_PLM_SP(n)_LONG_CS_TX1)

This software-initiated sequence immediately recovers both ends of the link from both input and output error-stopped conditions. Writing a value of 0x2003F044 into the Port n Control Symbol Transmit register causes a PNA and link request to be sent in the Stype 0 and 1 fields of control symbol to the link partner. The PNA causes the link partner to issue a link request to the DSP, and the link request causes the link partner to issue a link response. The DSP receiving the link request issues a link response. Figure B-1 illustrates the worst-case situation, where both device A and B are both in input and output error-stopped state. It is really only required to issue the Port n Control Symbol generation for one end of the link.

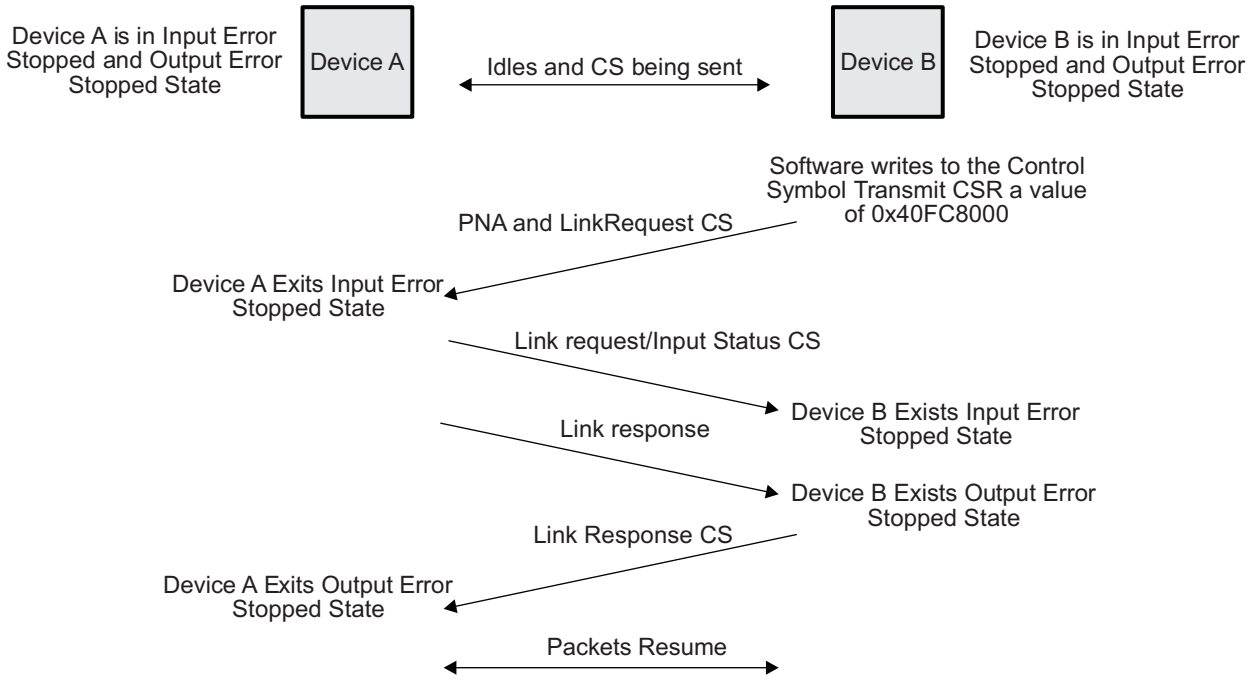


Figure B-1. Error Recovery Sequence Diagram

The above sequence causes both devices to exit all input and output error stopped states. However, it does not align ACKIDs, which must be synchronized before data or maintenance packets are sent between the link partners. If both link partners are coming out of reset, the ACKIDs will already be aligned and no additional steps are needed. If not, then additional steps must be immediately taken to align ACKIDs. TI's DSP supports the software error recovery registers which make this process fairly straight forward. The following steps can be used to align the ACKIDs if both link partners support these registers.

1. Write the RIO_PLM_SP(n)_LONG_CS_TX1 register = 0x2003F044 to exit the error states.
2. Each device's SP(n)_LM_RESP register will contain the link response data from step number 1. This data indicates the attached link partner's expected inbound ACKID value.
3. The DSP software can read the SP(n)_LM_RESP register and copy the expected partner's inbound ACKID to its own outbound and outstanding ACKID values in the local SP(n)_ACKID_STAT register.
4. The DSP can then send a maintenance packet to the link partner's SP(n)_ACKID_STAT register to program the ACKIDs to match expected values of the DSP. When the maintenance packet is sent, it overwrites all the fields of this register, so the value written should be: outstanding = outbound = DSP's expected inbound ACKID value, and the inbound = (1 + ACKID) & 0x1F from step 3.
5. ACKIDs are now aligned and data packets can be exchanged normally.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from November 1, 2010 to April 30, 2019 **Page**

- Removed Support for IDLE2 sequences. 24
- Changed parallel bus rate from 1/10th to 1/20th. 28
- Added Clock Summary section. 29
- Switched steps one and two in Writing to LSU_SETUP_REG0 list. 42
- Updated Error Handling section. 55
- Changed LOG_TGT_ID_DIS offset from 0x0020 to 0x0014. 86
- Changed LOG_TGT_ID_DIS offset from 0x0020 to 0x0014. 87
- Changed register at offset 1004h from TD_FDQUEUE_CTRL_REG to PERF_CTRL_REG. 133
- Updated SerDes Transmit Channel Configuration Register nn Descriptions table. 159
- Changed Reserved bits from 31-9 to 31-16 and 11-10. 167
- Added TXU_RETRY_TIMER_MODE to PER_SET_CNTL1 register. 167
- Updated SYS_CLK_VBUSP description in the PER_SET_CNTL1 register. 167
- Updated MTU bit field in DS_LL_CTL register. 234
- Updated CTRL_CAPT Register description. 262
- Updated PLM Port(n) Discovery Timer Register (Address offset 0x1B0B4, 0x1B1B4) table. 289
- Updated Port n Silence Timer (Address offset 0x1B0B8, 0x1B138, 0x1B1B8, 0x1B238) table. 290
- Updated Port IP Prescalar for IP_CLK Register (RIO_PRESCALAR_SRV_CLK) section. 352
- Added Appendix B: Software-Assisted Error Recovery. 366

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265