IBM PowerNP™
NP4GS3

Databook

Preliminary

Printed in the United States of America September 2000

Note: This document contains information on products in the design, sampling and/or initial production phases of development. This information is subject to change without notice. Verify with your IBM field applications engineer that you have the latest version of this document before finalizing a design.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction,
NY 12533-6351

The IBM home page can be found at
http://www.ibm.com

The IBM Microelectronics Division home page
can be found at http://www.chips.ibm.com

np3_DL_title.fm.01
09/25/00

Contents

List of Tables

List of Figures

About This Book

This databook describes the IBM PowerNP NP4GS3 and explains the basics of building a system using it.

A list of terms and abbreviations is provided in section *15. Glossary of Terms and Abbreviations* on page 433.

Who Should Read This Manual

This document is intended to provide information to programmers and engineers using the NP4GS3 for development of interconnect solutions for Internet or enterprise network providers.

Technology information to enable development of cards and boards includes electrical specifications and interface protocol and timings.

Programmer information includes descriptions of the Dyadic Protocol processors and the available command set. Additional information on configuring the network processor features for operation such as the egress scheduler and the flow control hardware is provided.

Related Publications

PPC405GP Embedded Controller User's Manual (http://www.chips.ibm.com/techlib/products/powerpc/datasheets.html)

PCI Specification, version 2.2. (http://www.pcisig.com)

Conventions Used in This Manual

The following conventions are used in this manual.

1. The bit notation for Physical Description, Physical MAC Multiplexer, Enqueuer / Dequeuer / Scheduler, Switch Interface, Embedded Processor Complex, Control Access Bus, and the Serial/Parallel Manager Interface, is non-IBM, meaning that bit zero is the least significant bit and bit 31 is the most significant bit in a 4-byte word.

   The bit notation for Tree Search Engine and Embedded PowerPC is IBM-standard, meaning that bit 31 is the least significant bit and bit zero is the most significant bit in a 4-byte word.

2. Nibble numbering is the same as byte numbering. The left-most nibble is most significant and starts at zero.

3. Overbars, e.g. $\overline{\text{TxEnb}}$, designate signals that are asserted "low".

4. Numeric notation is as follows:
   - Hexadecimal values are preceded by x or X. For example: x'0B00'.
   - Binary values in text are either spelled out (zero and one) or appear in quotation marks. For example: '10101'.

- Binary values in the Default and Description columns of the register sections are often isolated from text as in this example:

  0: No action on read access

  1: Auto-reset interrupt request register upon read access

5. Field length conventions are as follows:
   - 1 byte = 8 bits
   - 1 word = 4 bytes
   - 1 double word (DW) = 2 words = 8 bytes
   - 1 quadword (QW) = 4 words = 16 bytes
6. For signal and field definitions, when a field is designated as reserved ('r'):

   - As an input to the NP4GS3 it must be sent as zero.
   - As an output from the NP4GS3 it must not be checked or modified.
   - Its use as code point results in unpredictable behavior.

1. General Information

1.1 Features

- 4.5 million packets per second (Mpps) Layer 2 and Layer 3 Switching.
- 40 Fast Ethernet / 4 Gb MACs accessed through SMII, GMII, and TBI interfaces supporting industry standard PHY components. The Ethernet MAC provides 36 Ethernet statistics counters. Software can define an additional one million counters. Once these counts are defined, the hardware assists in updating them. With this hardware support, many standard MIBs can be supported at wire speed.
  - Supports IEEE 802.3 ad link aggregation and VLAN detection (frame type 8100).
  - 16xOC-3c / 4xOC-12 / 4xOC-12c / 1xOC-48 / 1xOC-48c integrated Packet over SONET (POS) interfaces support industry standard POS framers.
- Two Data-Aligned Synchronous Link (DASL) ports, rated 3.25 to 4 Gbps, for attachment to the Packet Routing Switch, another IBM PowerNP NP4GS3, or to itself through a wrap path for a stand-alone solution. (DASLs are electrically I/O compliant with the EIA/JEDEC JESD8-6 standard for differential HSTL.)
- Addressing capability of 64 target network processors, allowing the design of a network interconnect solution supporting 1024 ports.
- Advanced flow control mechanisms which tolerate high rates of temporary oversubscription without TCP collapse.
- Geometric hash functions yield lower collision rates than conventional bit scrambling methods, providing faster lookups and more powerful search engines.
- Hardware support for Port Mirroring[1]. Depending on the application, mirrored traffic can either share bandwidth with user traffic or use a separate Switch data path, eliminating the penalty normally associated with Port Mirroring.
- Support for jumbo frames (9018 without VLAN, 9022 with VLAN).
- Hardware managed and software configured bandwidth allocation control of 2048 concurrent communication flows.
- Embedded PowerPC™ and external 33/66 Mhz 32-bit PCI Bus for enhanced design flexibility.
  - Supports RISCWatch through the JTAG interface
- Eight Dyadic Protocol Processor Units (DPPU)
  - two picocode engines per DPPU
  - eight coprocessor units per DPPU to reduce picocode path lengths for common tasks
  - Multi-thread support of four threads per DPPU (two per picocode engine)
  - Zero context switching overhead between threads
- Serial management interface to support physical layer devices, board and box functions
- IBM SA-27E, 0.18 µm technology.
- Voltage ratings.
  - 1.8 V supply voltage
  - 2.5 V and 3.3 V compatibility with drivers and receivers
  - 1.25 V reference voltage for SSTL drivers
  - 1.5 V compatibility for DASL interfaces
- 1088-pin Bottom Surface Metallurgy - Ceramic Column Grid Array (BSM-CCGA) package with 815 Signal I/O.
- IEEE 1149.1a JTAG compliant.

1. OC48c ports are not supported by port mirroring functions.

1.2 Ordering Information

Part Number	Description
IBM32NPR161EPXCAC133	IBM PowerNP NP4GS3

1.3 Overview

The IBM PowerNP™ NP4GS3 is IBM's latest technology for supporting media-rate, multi-layer Ethernet switching. It also supports the IP over Sonet (POS) and PPP protocol. The NP4GS3 provides a highly customizable, scalable technology for the development of interconnect solutions for Internet or enterprise network providers. A single device can be used in desktop solution or be a component in a large multi-rack solution with up to 1024 ports. When used with the IBM Packet Routing Switch, addressing is limited to 640 ports. Scaling of this nature is accomplished through the use of IBM's high performance, non-blocking, packet switching technology and IBM's Data-Aligned Synchronous Link (DASL) interface which can be adapted to other industry switch technologies.

The NP4GS3 integrates switching engine, search engine, and security functions on one device to support the needs of customers who require high capacity, media rate switching of Layer 2 and 3 frames. Three switch priority levels are supported for port mirroring, high priority user frames, and low priority frames. The device's ability to enforce hundreds of rules with complex range and action specifications, a new industry benchmark for filtering capabilities, makes an NP4GS3-based system uniquely suited for server farm applications.

The NP4GS3 contains dyadic protocol processor units that work with hardware accelerators to support high speed pattern search, data manipulation, internal chip management functions, frame parsing, and data prefetching.

Systems developed with the NP4GS3 use a distributed software model. A rich instruction set includes conditional execution, packing (for input hash keys), conditional branching, signed and unsigned operations, counts of leading zeros, and more. To support this model, the device hardware and Code Development Suite include on-chip debugger facilities, a picocode assembler, and a picocode and system simulator which decrease the time to market for new applications.

The NP4GS3's scalability allows the design of multiple system configurations:

- Low-end systems with only one device which can use the device's switch interface to wrap traffic from the Ingress side to the Egress side.
- Medium-end systems with two devices which are directly interconnected through their switch interfaces.
- High-end systems with up to 64 NP4GS3s which are interconnected through a single or redundant switch. The IBM Packet Routing Switch is limited to addressing up to 16 NP4GS3s.

1.4 NP4GS3-Based System Design

The NP4GS3 contains eight dyadic protocol processor units (DPPUs) with 16 K words of internal picocode instruction store, providing 2128 MIPS of processing power. Each DPPU contains two processors (CLPs) which share eight dedicated coprocessors. All eight DPPUs share three hardware accelerators. Each CLP can run two threads; each DPPU can therefore run four threads. Context switching occurs when the CLP is waiting for a shared resource (for example, waiting for one of the coprocessors to complete an operation, return the results of a search, or access DRAM). Context switching is accomplished by hardware assists that maintain separate sets of General Purpose Registers (GPR). Coprocessors can run in parallel with the protocol processors.

The Control Store for the protocol processors is provided by both internal and external memories: internal SRAM for immediate access, external ZBT SRAM for fast access, and external DDR SDRAM for large storage requirements. The internal Control Access Bus (CAB) allows access to internal registers, counters, and memory. The CAB also includes an external interface to control instruction step and interrupt control for debugging and diagnostics.

1.4.1 Coprocessors

- The Data Store coprocessor interfaces frame buffer memory (ingress and egress directions) providing a 320-byte working area.
- The Checksum coprocessor calculates and verifies header checksums.
- The Enqueue coprocessor manages control blocks containing key frame parameters. This coprocessor interfaces with the Completion Unit hardware assist, to enqueue frames to the switch and target port output queues.
- The CAB interface coprocessor controls thread access to the CAB through the CAB Arbiter.
- The String Copy coprocessor accelerates data movement between coprocessors.
- The Counter coprocessor manages counter updates for the picocode engines.
- The Policy coprocessor determines if the incoming data stream complies with configured profiles.
- The Tree Search Engine performs pattern analysis through tree searches, along with regular read and write accesses, all protected by memory range checking.

1.4.2 Hardware Accelerators

- The Completion Unit assures frame order.
- The Dispatch Unit parses the work out among the dyadic processors.
- The Control Store Arbiter is a shared memory interface.

1.4.3 NP4GS3 Memory

The NP4GS3 utilizes:

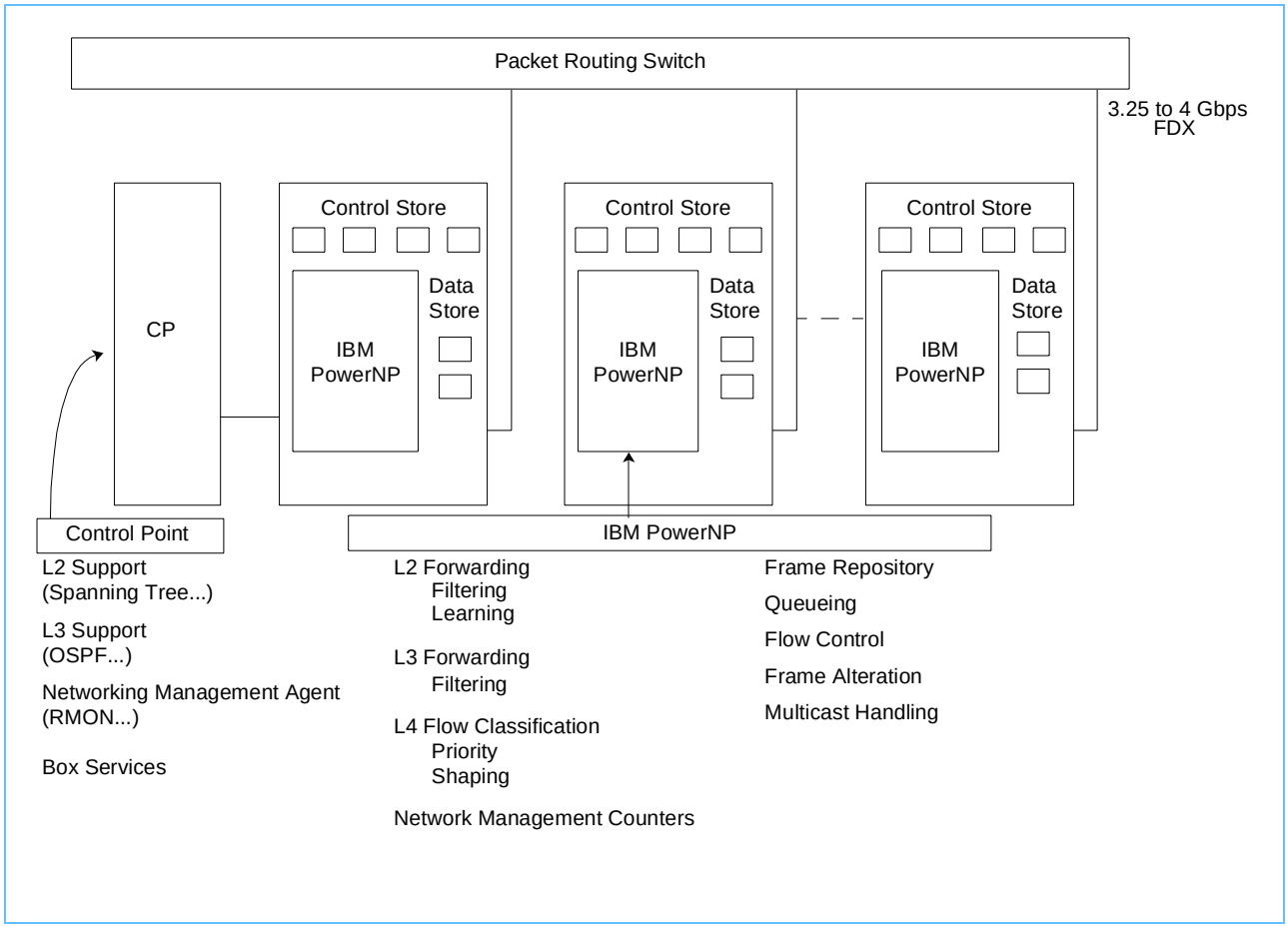
- A common instruction memory which holds 16 K instruction words for normal processing and control functions
- A 128 K byte internal SRAM for input frame buffering
- 113 K bytes of internal SRAM Control Store
- High capacity external DDR DRAM for egress frame buffering and to support large forwarding tables
- External ZBT SRAM for fast table access

1.4.4 Distributed Software Model

Systems developed with the NP4GS3 use a distributed software model. The system relies on a control point function to provide support for Layer 2 and Layer 3 routing protocols, Layer 4 and Layer 5 network applications, and systems management. The control point function may be provided by an external microprocessor connected through industry standard full duplex Ethernet links using either Fast or Gigabit Ethernet. However, control point functions in a smaller system configuration can be performed by the device's Embedded PowerPC processor. Forwarding and filtering tables created by the control point function are downloaded to the Control Store using Guided Frames. Guided Frames are the in-band control path between the control

point function and all network processor devices in a the system. Other functions, such as forwarding, filtering and classification, are performed at wire speed in a distributed way by the hardware and resident picocode of each device in the system.

Figure 1: Function Placement in an NP4GS3-Based System



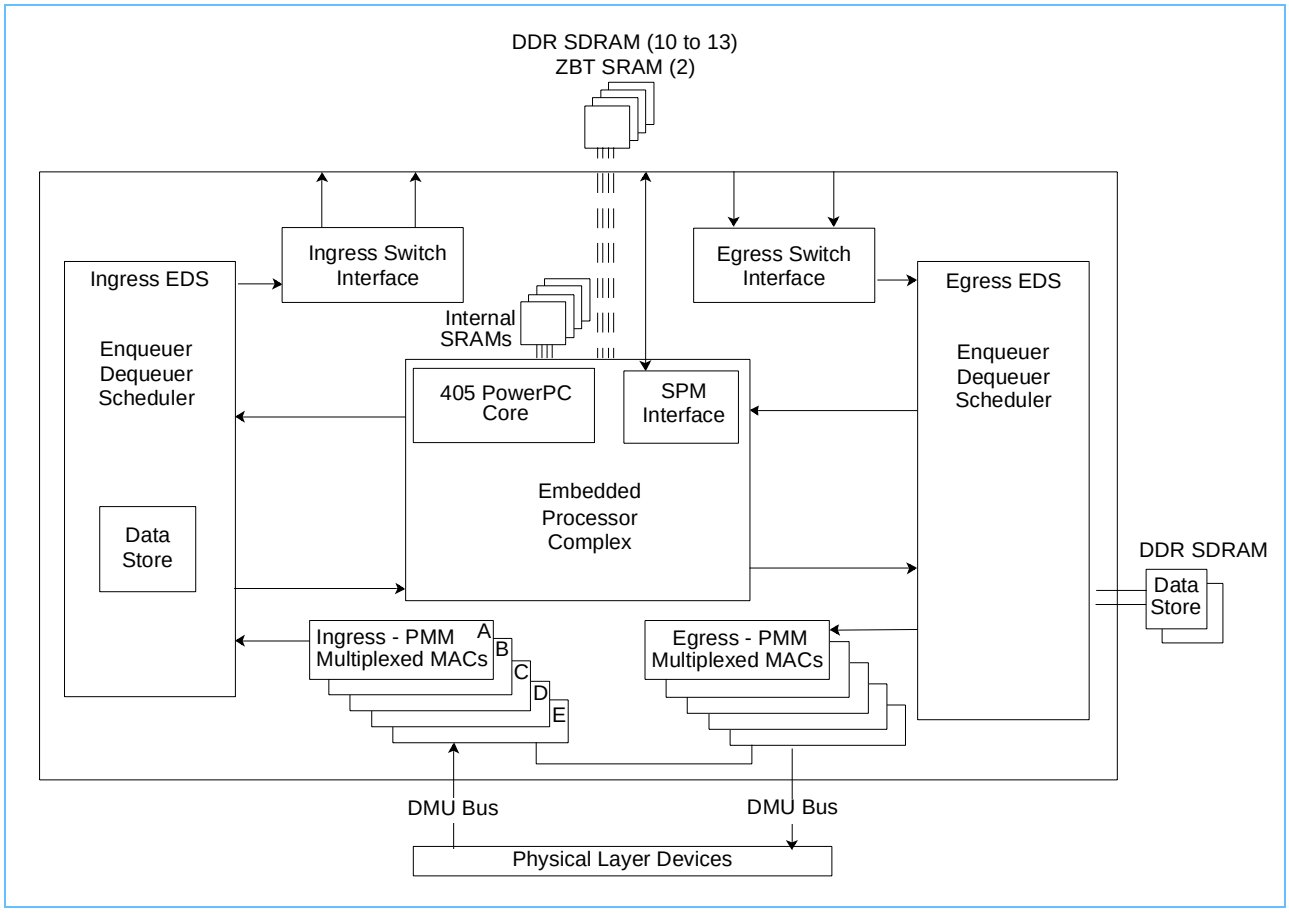
1.5 NP4GS3 Structure

The NP4GS3 has eight major functional blocks:

- Embedded Processor Complex (EPC)
- Embedded 405 PowerPC Core. The Control Store interface provides for 64 MB program space for the Power PC
- Enqueuer / Dequeuer / Scheduler logic for frames traveling from the physical layer devices to the switch fabric (Ingress-EDS)
- Enqueuer / Dequeuer / Scheduler logic for frames traveling from the switch fabric to the physical layer devices (Egress-EDS)
- Ingress Switch Interface (Ingress-Switch)
- Egress Switch Interface (Egress-Switch)
- Physical MAC Multiplexer (Ingress-PMM) receiving frames from the physical layer devices (Ethernet, POS framers, etc.)

- Physical MAC Multiplexer (Egress-PMM) transmitting frames to the physical layer devices

Figure 2: NP4GS3 Major Sections



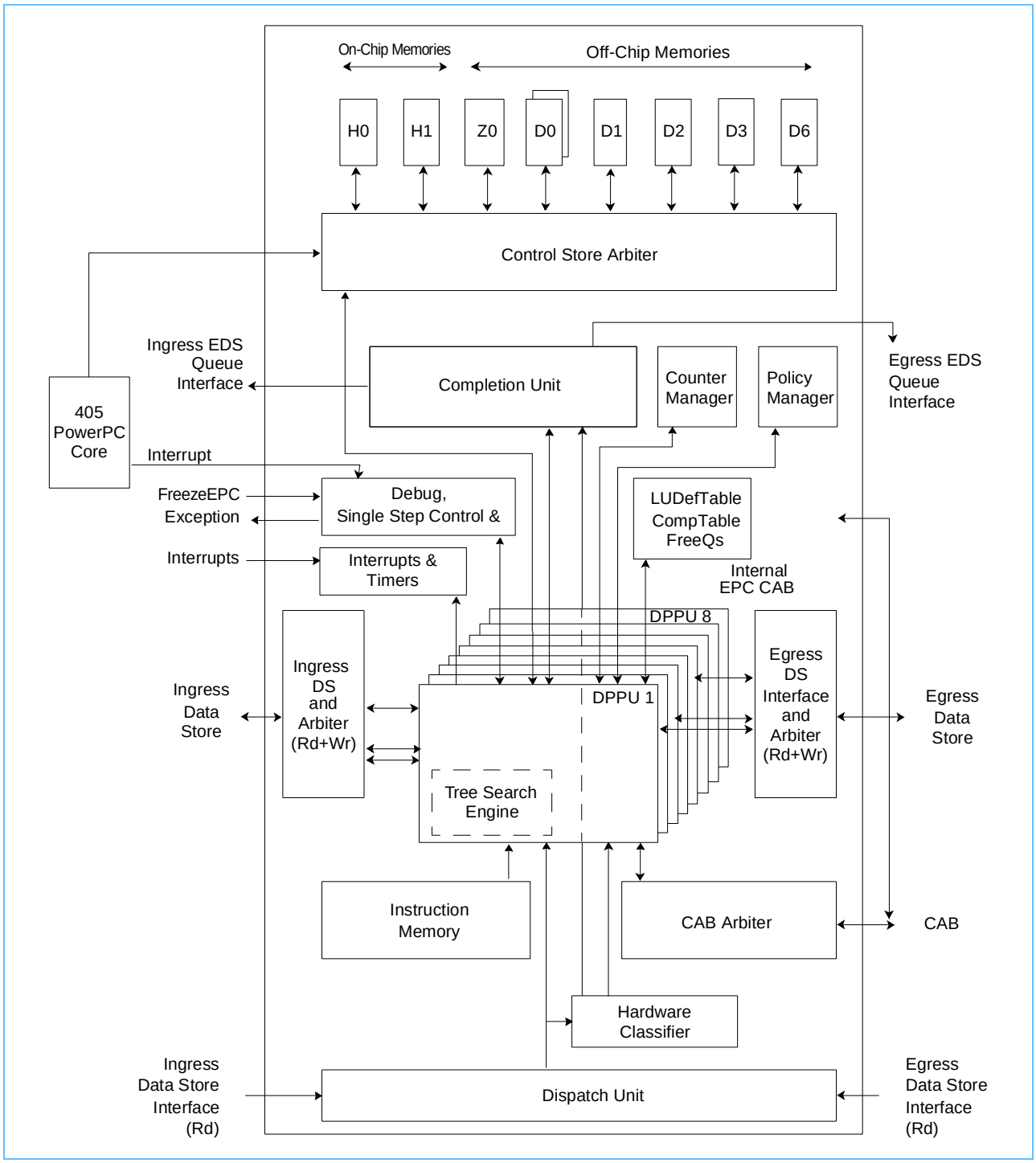
1.6 NP4GS3 Data Flow

The Ingress EDS places frames received from a physical layer device into the Ingress Data Store. These frames are then identified as either normal Data Frames or system control Guided Frames and enqueued to the EPC. The EPC contains eight dyadic protocol processor units capable of operating on up to 32 frames in parallel. Three of the threads within the eight dyadic protocol processors are enhanced: one for handling Guided Frames (the Guided Frame Handler or GFH), one for building table data in Control Memory (the General Table Handler or GTH), and one for handling communication with the embedded PowerPC.

The EPC also contains the following:

- A Dispatch Unit to dispatch new frames to idle threads
- A Completion Unit to maintain frame sequence
- Common Instruction Memory
- Hardware Classifier to parse the frame on-the-fly, preparing its processing by picocode
- Ingress and Egress Data Store interfaces to control read and write operations of data buffers
- Control Store Arbiter to allow the processors to share access to the Control Memory
- CAB Control, CAB Arbiter, and CAB Interface to allow debug access to NP4GS3 data structures

Figure 3: Embedded Processor Complex Block Diagram



Data frames are dispatched by the Dispatch Unit to the next available protocol processor for performing frame lookups, filtering, etc. Frame data is passed to the protocol processor along with results from the Hardware Classifier (HC). The HC parses bridged, IP, and IPX frame formats and passes the results to the proto-

col processor. The results determine the Tree Search algorithm and the starting Common Instruction Address (CIA). Tree Search algorithms supported are:

- Full Match Trees (fixed size patterns requiring exact match, such as Layer 2 Ethernet MAC tables)
- Longest Prefix Match Trees (patterns requiring variable length matches, such as subnet IP forwarding)
- Software Managed Trees (patterns defining a range or a bit mask set, such as those used for filter rules)

The Tree Search Engine (TSE) performs table searches. The TSE performs Control Store accesses independently, freeing the protocol processor for parallel execution. The Control Store contains all tables, counters, and any other data needed by the picocode. The Control Store Arbiter manages all Control Store operations, allocating memory bandwidth among the threads of the protocol processors.

Frame data is accessed through the Data Store coprocessor, which manages a 320-byte data buffer and control blocks for Data Store operations. Ingress frame alterations, such as VLAN header insertion or overlay, are defined once the forwarding requirements are found. These alterations are not performed by the Embedded Processor Complex. Instead, hardware flags are associated with the frame and Ingress Switch Interface hardware performs alterations while moving data. Other frame alterations can be accomplished by the picocode and the Data Store coprocessor by directly modifying frame contents held in the Ingress Data Store.

Control data is gathered and used to build Switch Headers and Frame Headers prior to sending frames to the switch fabric. Control data includes switch information, such as the destination switch port of the frame, and information for the Egress side of the target device to help the device expedite frame lookup of destination ports and multicast or unicast operations, and determine necessary Egress frame alterations.

Upon completion, the Enqueue coprocessor builds the necessary information to enqueue the frame to the switch interface and provides it to the Completion Unit (CU). The CU guarantees the frame order from the 32 threads to the switch interface queues. Frames from the switch interface queues are segmented into 64-byte cells with Cell Header and Frame Header bytes inserted as they are transmitted to the Switch Interface.

Frames received from the switch interface are placed in Egress Data Store (E-DS) buffers by the Egress EDS and presented to the EPC. The Dispatch Unit fetches a portion of the frame and delivers it to an available dyadic protocol processor to perform frame forwarding. Frame data is dispatched to the dyadic protocol processor along with data from the HC. The HC uses frame control data created by the Ingress device to help determine the beginning CIA.

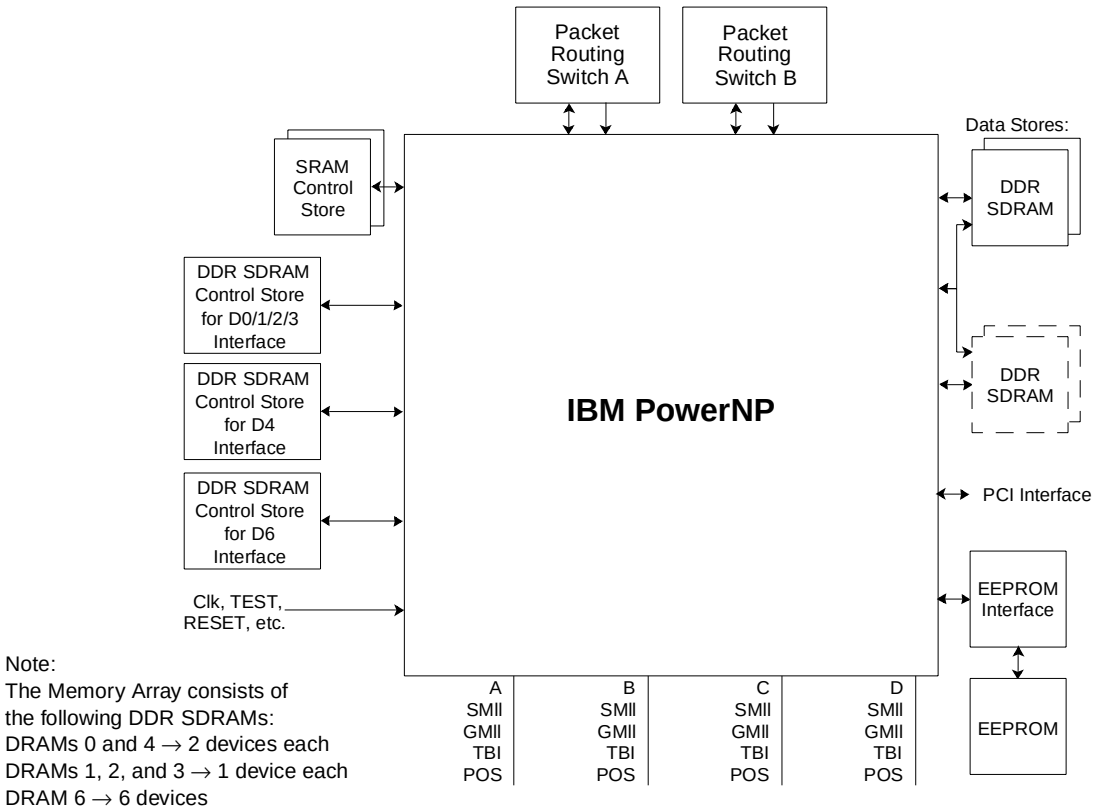
Egress table searches support the same algorithms as those supported for Ingress searches. The TSE performs table searches, freeing the protocol processor to continue parallel execution. All Control Store operations are managed by the Control Store Arbiter, which allocates memory bandwidth among the threads of all the dyadic protocol processors.

Egress frame data is accessed through the Data Store coprocessor. The result of a successful lookup contains forwarding information and, if needed, frame alteration information. Egress frame alterations can include VLAN header deletion, Time To Live increment (IPX) or decrement (IP), IP Header Checksum recalculation, Ethernet frame CRC overlay, and MAC DA/SA overlay or insertion. These alterations are not performed by the Embedded Processor Complex. Rather, hardware flags are associated with the frame, and Egress PMM hardware performs the alterations while moving data. Upon completion, the Enqueue coprocessor is used to build the necessary information for enqueueing the frame in the Egress EDS queues and provides it to the Completion Unit. The Completion Unit guarantees frame order for the eight dyadic protocol processor units' 32 threads to the Egress EDS queues feeding the egress ports.

The completed frames are then sent by Egress PMM hardware to the attached physical layer devices.

2. Physical Description

Figure 4: Devices Interfaces



Note:
The Memory Array consists of the following DDR SDRAMs:
DRAMs 0 and 4 → 2 devices each
DRAMs 1, 2, and 3 → 1 device each
DRAM 6 → 6 devices

2.1 Pin Information

The tables listed below (found on pages 32–63) describe the device pins arranged by functionality:

- **IBM 28.4 Gbps Packet Routing Switch Interface Pins** interface between the NP4GS3 and the Packet Routing Switch fabric.
- **Flow Control Pins**
- **Z0 ZBT SRAM Interface Pins** interface to the Z0 Zero Bus Turnaround (ZBT) SRAM for lookup.
- **Z1 ZBT SRAM Interface Pins**
- **D3, D2, and D1 Memory Pins and D0 Memory Pins** interface to the double-data rate (DDR) SDRAM used to implement the D3, D2, D1, and D0 memories.
- **D6_5 / D6_4 / D6_3 / D6_2 / D6_1 / D6_0 Memory Pins** interface to the double-data rate (DDR) SDRAM used to implement the PowerPC Store.
- **D4_0 / D4_1 Memory Pins** interface to the double-data rate (DDR) DRAM to implement the D4 memories.
- **DS1 and DS0 Pins** interface to the double-data rate (DDR) DRAM used to implement the DS1 and DS0 memories.
- **PMM Interface Pins** interface to the PHY (TBI bus, SMII bus, GMII bus, POS bus
- **PCI Interface Pins** interface to the PCI bus
- **Management Bus Pins** are translated into various “host” buses by an external FPGA (SPM).
- **Miscellaneous Pins**

For information on signal pin locations, see *Table 32: Complete Signal Pin Listing by Signal Name* on page 71 and *Table 33: Complete Signal Pin Listing by Grid Position* on page 81.

Table 1: I/O Signal Pin Summary

Interface	Quantity	Interface	Quantity
DASL-A	32	DMU-A	31
DASL-B	32	DMU-B	31
Master Grant (Master_Grant) A and B	4	DMU-C	31
Multicast Grant (MC_Grant) A and B	4	DMU-D	31
Send Grant (Send_Grant) A and B	2	Rx_Byte(1:0)	2
Ingress Free Queue Threshold (I_FreeQ_Th)	1	PCI	52
Flow Control Pins [RES_Sync, RES_Data]	2	Management Bus	3
ZBT0	57	2x Switch Clock Differential	4
ZBT1	39	DASL A/B select (Switch_BNA)	1
D1 / D2 / D3 Shared (Shared among five DDR SDRAMs)	6	Core_Clock (53.3 Mhz)	1
D3_0	33	125 MHz Clock	1
D2_0	33	$\overline{\text{RESET}}$	1
D1_0	33	Not operational	1
D0 Shared (Clk, $\overline{\text{Clk}}$, RAS, CAS, BA(1:0))	6	Test Mode	2
D0_0/0_1	51	RISCWatch/JTAG	6
D4_0/D4_1 (Clk, $\overline{\text{Clk}}$, RAS, CAS, BA(1:0)) (Shared among two DDR SDRAMs)	6	PLL_Analog V_{DD}	3
D4_0/D4_1	51	PLL_Analog GND	3
D6 (Clk, $\overline{\text{Clk}}$, RAS, CAS, BA(1:0)) (Shared among six DDR SDRAMs)	6	Thermal (_In/_Out)	2
D6_0/D6_1/D6_2/D6_3/D6_4/D6_5	41	VRef	12
DS0_0/0_1/DS1_0/1_1 Shared (Clk, $\overline{\text{Clk}}$, RAS, CAS, BA(1:0)) (Shared among four DDR SDRAMs)	6	Code Boot strapping (Boot_Picocode, Boot_PPC)	2
DS1_0/_1	51	Spare test receivers	10
DS0_0/1	51		
Total used			777
Unused[1]			38

1. All unused pins should be left unconnected on the card.

Table 2: IBM 28.4 Gbps Packet Routing Switch Interface Pins

Signal (Clock Domain)	Description	Type
DASL_Out_A(7:0) (Switch Clk * 8)	The positive half of an output bus of eight custom low power differential drivers. Runs at frequency Switch_Clock_A * 8.	Output DASL 1.5 V
$\overline{\text{DASL\_Out\_A(7:0)}}$ (Switch Clk * 8)	The negative half of the 8-bit differential bus described above. Runs at frequency Switch_Clock_A * 8.	Output DASL 1.5 V
DASL_In_A(7:0) (Switch Clk * 8)	The positive half of an input bus of eight custom low power differential receivers. Runs at frequency Switch_Clock_A * 8.	Input DASL 1.5 V
$\overline{\text{DASL\_In\_A(7:0)}}$ (Switch Clk * 8)	The negative half of the 8-bit differential bus described above. Runs at frequency Switch_Clock_A * 8.	Input DASL 1.5 V
DASL_Out_B(7:0) (Switch Clk * 8)	The positive half of an output bus of eight custom low power differential drivers. Runs at frequency Switch_Clock_B * 8.	Output DASL 1.5 V
$\overline{\text{DASL\_Out\_B(7:0)}}$ (Switch Clk * 8)	The negative half of the 8-bit differential bus described above. Runs at frequency Switch_Clock_B * 8.	Output DASL 1.5 V
DASL_In_B(7:0) (Switch Clk * 8)	The positive half of an input bus of eight custom low power differential receivers. Runs at frequency Switch_Clock_B * 8.	Input DASL 1.5 V
$\overline{\text{DASL\_In\_B(7:0)}}$ (Switch Clk * 8)	The negative half of the 8-bit differential bus described above. Runs at frequency Switch_Clock_B * 8.	Input DASL 1.5 V
Master_Grant_A(1:0) (Switch Clk * 2)	Master Grant A indicates whether the “A” connection of the switch fabric is able to receive cells from the NP4GS3. The definitions of these I/Os are configured by the Master Grand mode configuration registers. See *Section 13.2* on page 344.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Master_Grant_B(1:0) (Switch Clk * 2)	Master Grant B indicates whether the “B” connection of the switch fabric is able to receive cells from the NP4GS3. The definitions of these I/Os are configured by the Master Grand mode configuration registers. See *Section 13.2* on page 344.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Multicast_Grant_A(1:0)	Multicast Grant A indicates whether the “A” connection of the switch fabric is able to receive multicast cells. 0 Unable 1 Able Bit 0 of this bus serves as the master grant for the high priority channel, and Bit 1 for the low priority channel. This signal runs at frequency Switch Clock * 2.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Multicast_Grant_B(1:0)	Multicast Grant B indicates whether the “B” connection of the switch fabric is able to receive multicast cells from the NP4GS3. 0 Unable 1 Able Bit 0 of this bus serves as the master grant for the high priority channel, and Bit 1 for the low priority channel. This signal runs at frequency Switch Clock * 2.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Send_Grant_A (Switch Clk * 2)	Send Grant A indicates whether the “A” connection of the NP4GS3 is able to receive cells from the switch fabric. 0 Unable (the Packet Routing Switch should send only idle cells) 1 Able The NP4GS3 changes the state of this signal.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 2: IBM 28.4 Gbps Packet Routing Switch Interface Pins (Continued)

Signal (Clock Domain)	Description	Type
Send_Grant_B (Switch Clk * 2)	Send Grant B indicates whether the “B” connection of the NP4GS3 is able to receive cells from the switch fabric. 0 Unable (the Packet Routing Switch should send only idle cells) 1 Able The NP4GS3 changes the state of this signal.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 3: Flow Control Pins

Signal	Description	Type
I_FreeQ_Th	Ingress Free Queue Threshold 0 Threshold not exceeded 1 Threshold exceeded	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
RES_Sync	Remote Egress Status synchronization (sync) is driven by the network processor that is configured to provide this signal. It is received by all other network processors. 1 Shared data bus sync pulse. Indicates start of time division multiplex cycle.	Input/Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
RES_Data	Remote Egress Status data is driven by a single network processor during its designated time slot. 0 Not exceeded 1 Network processor’s exponentially weighted moving average (EWMA) of the egress offered rate exceeds the configured threshold.	Input/Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 4: Z0 ZBT SRAM Interface Pins

Signal	Description	Type
LU_Clk	Look-Up clock. 7.5 ns period (133 MHz).	Output CMOS 2.5 V
LU_Addr(18:0)	Look-Up Address signals are sampled by the rising edge of LU_Clk.	Output CMOS 2.5 V
LU_Data(35:0)	Look-Up Data. When used as SRAM inputs, the rising edge of LU_Clk samples these signals.	Input/Output CMOS 2.5 V
LU_R_$\overline{\text{Wrt}}$	Look-Up Read/Write control signal is sampled by the rising edge of LU_Clk. 0 Write 1 Read	Output CMOS 2.5 V

Table 5: Z1 ZBT SRAM Interface Pins

Signal	Description	Type
SCH_Clk	SRAM Clock input. 7.5 ns period (133 MHz).	Output CMOS 2.5 V
SCH_Addr(18:0)	SRAM Address signals are sampled by the rising edge of LU_Clk.	Output CMOS 2.5 V
SCH_Data(17:0)	Databus. When used as an SRAM input, the rising edge of SCH_Clk samples these signals.	Input/Output CMOS 2.5 V
SCH_R_$\overline{Wrt}$	Read/Write control signal is sampled by the rising edge of SCH_Clk. 0 Write 1 Read	Output CMOS 2.5 V

Figure 5: ZBT SRAM Timing Diagram

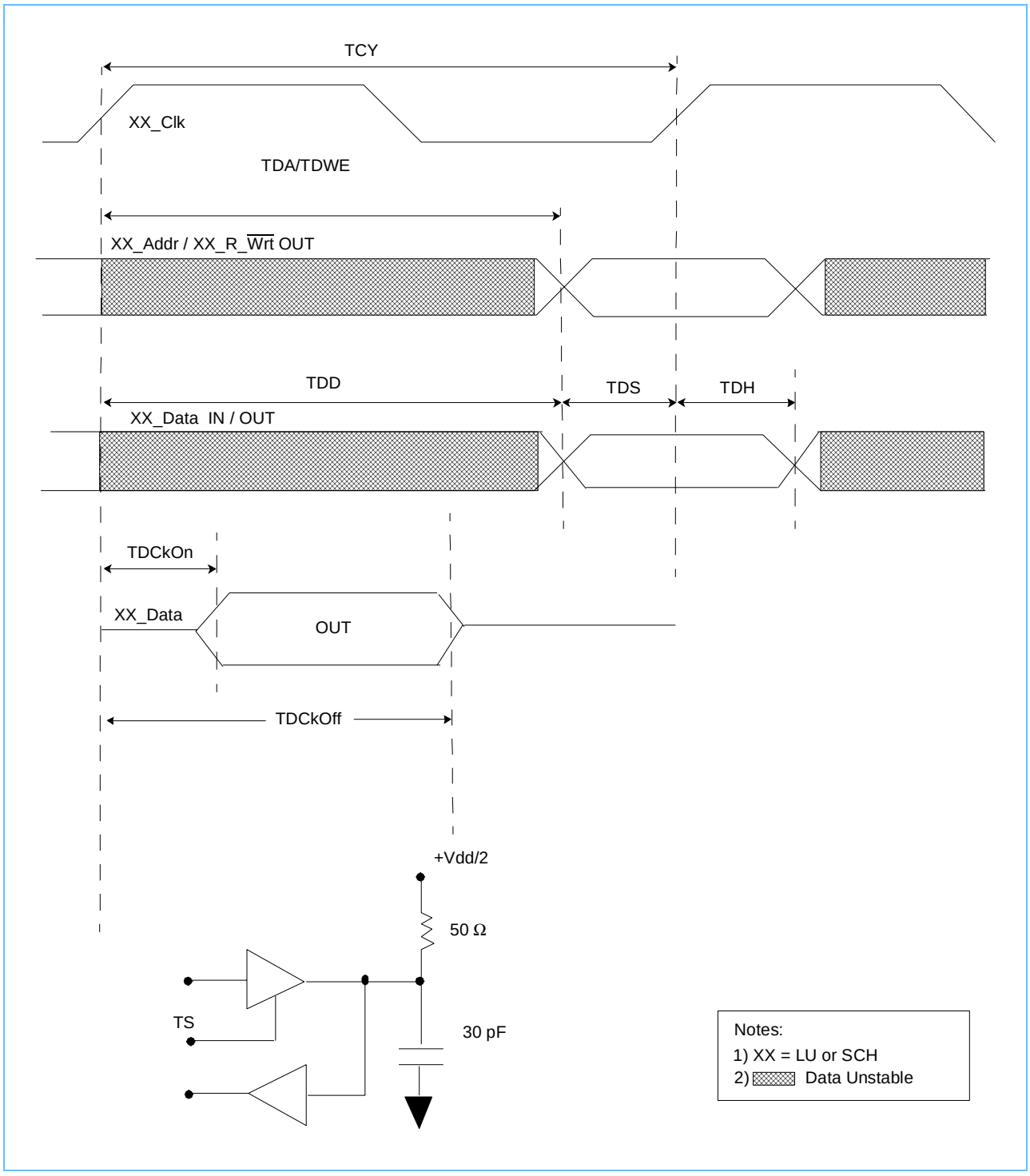


Table 6: ZBT SRAM Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	ZBT Cycle Time	7.5	
TDA	Address Output Delay[1]	2.28	4.25
TDWE	Read/Write Output Delay[1]	2.92	5.22
TDD	Data Output Delay[1]	1.75	4.98
TDCkOn	Data Output Turn On[2]	3.28	
TDCkOff	Data Output Turn Off		5.65
TDS	Input Data Setup Time	2.44	
TDH	Input Data Hold Time	0	

1. To determine delay from NP4GS3 internal clock, add 9.14 ns to Minimum Delay and 8.38 ns to Maximum Delay.
2. To determine delay from NP4GS3 internal clock, add 3.33 ns to Minimum Delay.
3. Blank space indicates no limit.
4. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 7: D3, D2, and D1 Memory Pins

Signal	Description	Type
Shared Signals		
DB_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D3, D2, and D1 memory devices.	Output SSTL2 2.5 V
$\overline{\text{DB\_Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the D3, D2, and D1 memory devices.	Output SSTL2 2.5 V
DB_RAS	Common row address strobe (common to D3, D2, and D1	Output SSTL2 2.5 V
DB_CAS	Common column address strobe (common to D3, D2, and D1	Output SSTL2 2.5 V
DB_BA(1:0)	Common bank address (common to D3, D2, and D1	Output SSTL2 2.5 V
D3 Signals		
D3_Addr(12:0)	D3 address	Output CMOS 2.5 V
D3_DQS(1:0)	D3 data strobes	Input/Output SSTL2 2.5 V
D3_Data(15:0)	D3 data bus	Input/Output SSTL2 2.5 V
D3_WE	D3 write enable	Output CMOS 2.5 V

Table 7: D3, D2, and D1 Memory Pins (Continued)

Signal	Description	Type
D3_CS	D3 chip select	Output CMOS 2.5 V
D2 Signals		
D2_Addr(12:0)	D2 address	Output CMOS 2.5 V
D2_DQS(1:0)	D2 data strobes	Input/Output SSTL2 2.5 V
D2_Data(15:0)	D2 data bus	Input/Output SSTL2 2.5 V
D2_WE	D2 write enable	Output CMOS 2.5 V
D2_CS	D2 chip select	Output CMOS 2.5 V
D1 Signals		
D1_Addr(12:0)	D1 address	Output CMOS 2.5 V
D1_DQS(1:0)	D1 data strobes	Input/Output SSTL2 2.5 V
D1_Data(15:0)	D1 data bus	Input/Output SSTL2 2.5 V
D1_WE	D1 write enable	Output CMOS 2.5 V
D1_CS	D1 chip select	Output CMOS 2.5 V

Table 8: D0 Memory Pins

Signal	Description	Type
D0_0 and D0_1 Shared Signals		
DE_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D0_0/1 memory devices.	Output SSTL2 2.5 V
DE_$\overline{\text{Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the D0_0/1 devices.	Output SSTL2 2.5 V
DE_RAS	Common row address strobe	Output CMOS 2.5 V

Table 8: D0 Memory Pins

Signal	Description	Type
DE_CAS	Common column address strobe	Output CMOS 2.5 V
DE_BA(1:0)	Common bank address	Output CMOS 2.5 V
D0_Addr(12:0)	D0 address	Output CMOS 2.5 V
D0_DQS(3:0)	D0 data strobes	Input/Output SSTL2 2.5 V
D0_Data(31:0)	D0 data bus	Input/Output SSTL2 2.5 V
D0_WE	D0 write enable	Output CMOS 2.5 V
D0_CS	D0 chip select	Output CMOS 2.5 V

Table 9: D4_0 / D4_1 Memory Pins

Signal	Description	Type
DD_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D4_0/1 memory devices.	Output SSTL2 2.5 V
DD_$\overline{\text{Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the D4_0/1 memory devices.	Output SSTL2 2.5 V
DD_RAS	Common row address strobe	Output CMOS 2.5 V
DD_CAS	Common column address strobe	Output CMOS 2.5 V
DD_BA(1:0)	Common bank address	Output CMOS 2.5 V
D4_Addr(12:0)	D4 address	Output CMOS 2.5 V
D4_DQS(3:0)	D4 data strobes	Input/Output SSTL2 2.5 V
D4_Data(31:0)	D4 data bus	Input/Output SSTL2 2.5 V

Table 9: D4_0 / D4_1 Memory Pins

Signal	Description	Type
D4_WE	D4 write enable	Output CMOS 2.5 V
D4_CS	D4 chip select	Output CMOS 2.5 V

Table 10: D6_5 / D6_4 / D6_3 / D6_2 / D6_1 / D6_0 Memory Pins

Signal	Description	Type
DA_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D6 memory chips.	Output SSTL2 2.5 V
DA_$\overline{\text{Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the D6 memory devices.	Output SSTL2 2.5 V
DA_RAS	Common row address strobe (common to D6)	Output SSTL2 2.5 V
DA_CAS	Common column address strobe (common to D6)	Output SSTL2 2.5 V
DA_BA(1:0)	Common bank address (common to D6)	Output SSTL2 2.5 V
D6_WE	Common write enable (common to D6)	Output SSTL2 2.5 V
D6_Addr(12:0)	D6 address	Output SSTL2 2.5 V
D6_CS	D6 chip select	Output SSTL2 2.5 V
D6_DQS(3:0)	D6 data strobes	Input/Output SSTL2 2.5 V
D6_Data(15:0)	D6 data bus	Input/Output SSTL2 2.5 V
D6_ByteEn(1:0)	D6 byte enables byte masking write to D6.	Input/Output SSTL2 2.5 V
D6_Parity(1:0)	D6 parity signals, one per byte. Must go to separate chips to allow for byte write capability.	Input/Output SSTL2 2.5 V
D6_DQS_Par(1:0)	D6 data strobe for the parity signals	Input/Output SSTL2 2.5 V

Table 11: DS1 and DS0 Pins

Signal	Description	Type
Shared Signals		
DC_Clk	The positive pin of an output differential pair. 133 MHz. Common to the DS1 and DS0 memory devices.	Output SSTL2 2.5 V
DC_$\overline{\text{Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the DS1 and DS0 memory devices.	Output SSTL2 2.5 V
DC_RAS	Common Row address strobe (common to DS1 and DS0)	Output SSTL2 2.5 V
DC_CAS	Common Column address strobe (common to DS1 and DS0)	Output SSTL2 2.5 V
DC_BA(1:0)	Common bank address (common to DS1 and DS0)	Output SSTL2 2.5 V
DS1 Signals		
DS1_Addr(12:0)	DS1 address	Output CMOS 2.5 V
DS1_DQS(3:0)	DS1 data strobes	Input/Output SSTL2 2.5 V
DS1_Data(31:0)	DS1 data bus	Input/Output SSTL2 2.5 V
DS1_WE	DS1 write enable	Output CMOS 2.5 V
DS1_CS	DS1 chip select	Output CMOS 2.5 V
DS0 Signals		
DS0_Addr(12:0)	DS0 address	Output CMOS 2.5 V
DS0_DQS(3:0)	DS0 data strobes	Input/Output SSTL2 2.5 V
DS0_Data(31:0)	DS0 data bus	Input/Output SSTL2 2.5 V
DS0_WE	DS0 write enable	Output CMOS 2.5 V
DS0_CS	DS0 chip select	Output CMOS 2.5 V

Figure 6: DDR Timing Diagram

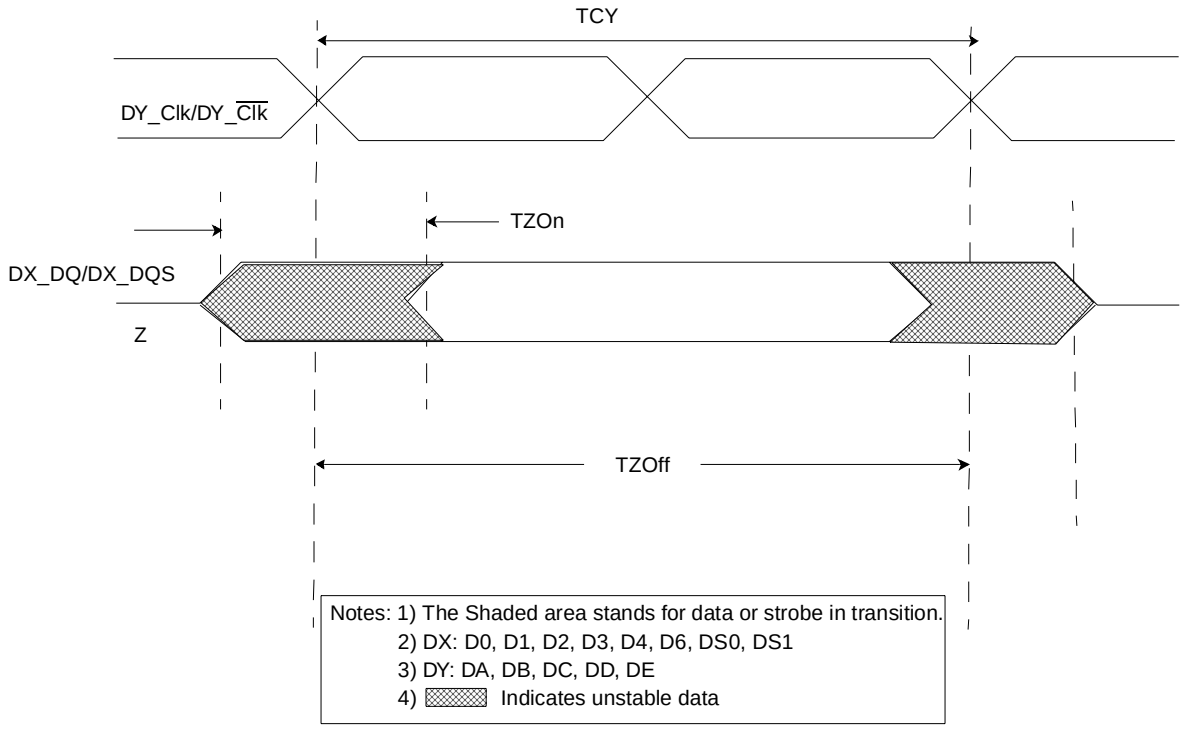


Figure 7: DDR Read Input Timing Diagram

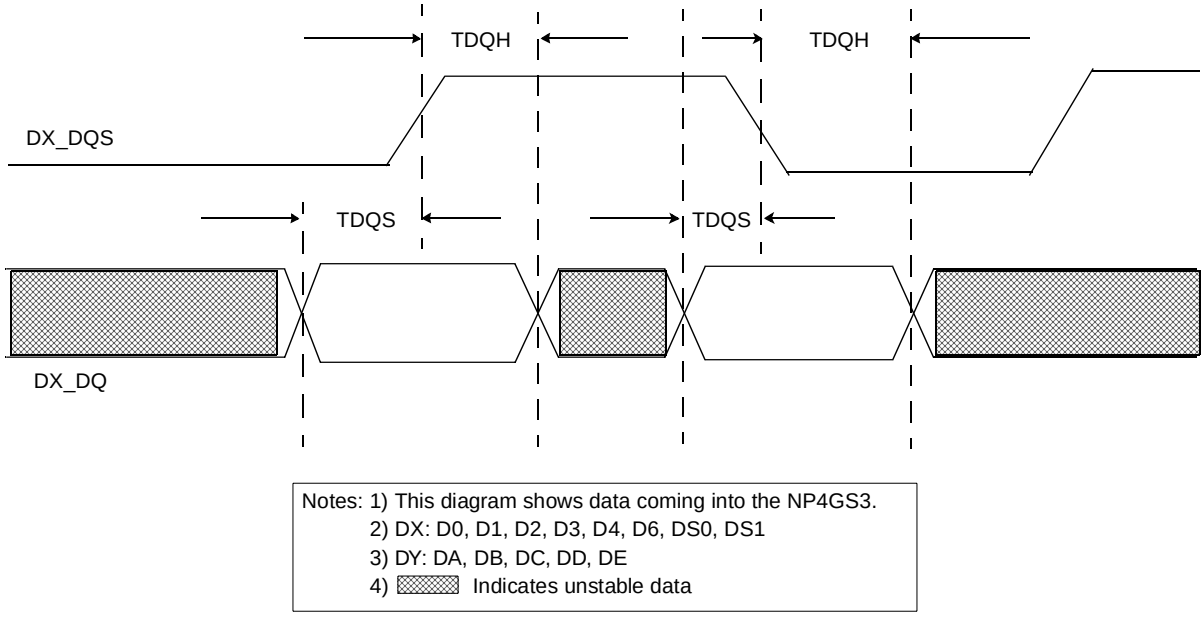


Figure 8: DDR Write Output Timing Diagram

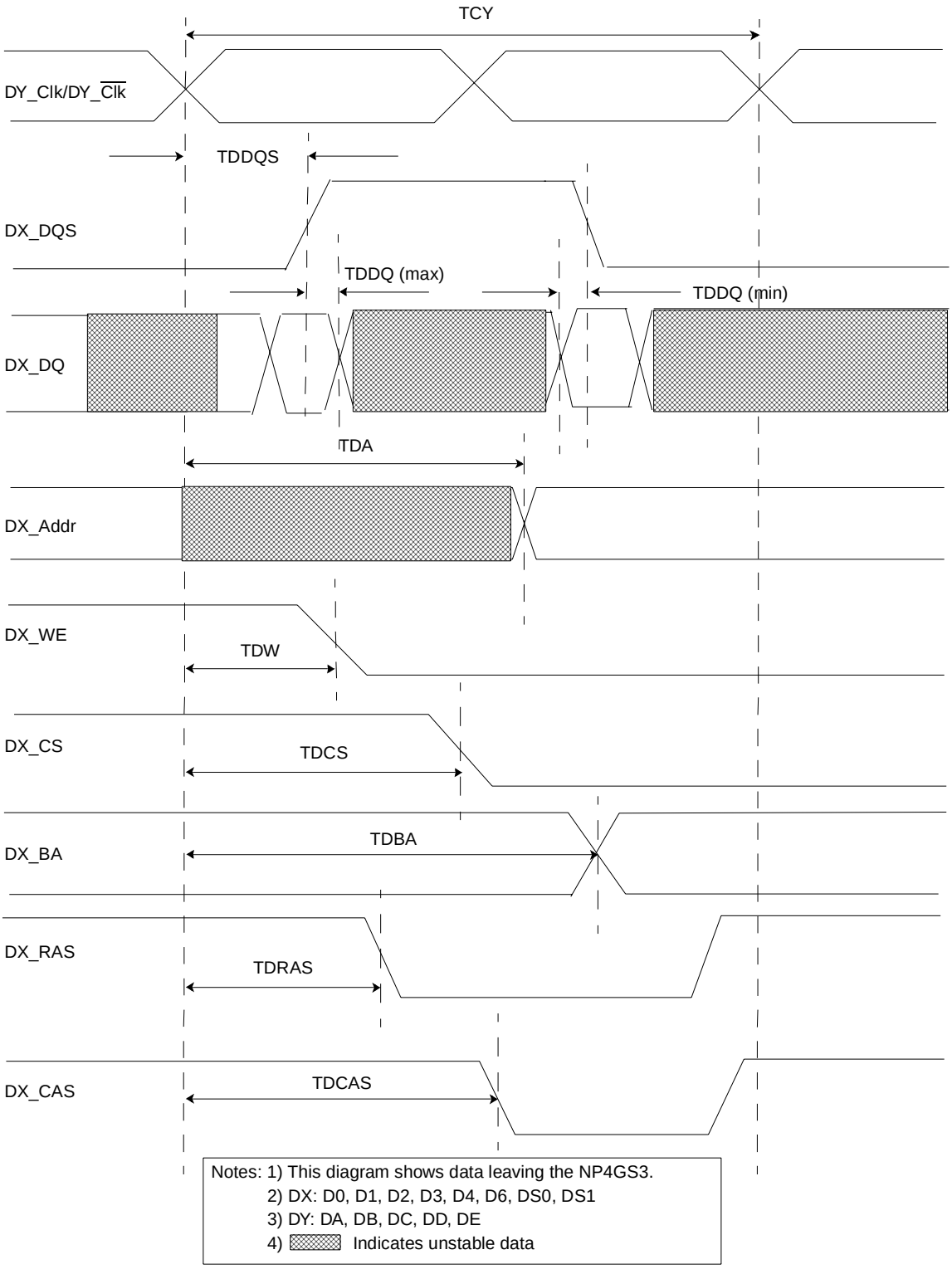


Table 12: DDR Timing Diagrams Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	DDR Clock Cycle Time	7.5	
TDDQS	DQS Output Strobe to Clock Output Delay	-0.9	1.29
TDDQ	DQ Data to DQS Skew	-2.2	2.45
TDA	Address Output Delay[1]	0.79	4.85
TDW	Write Enable Output Delay[1]	1.00	4.88
TDCS	Chip Select Output Delay[1]	0.93	5.62
TDBA	Bank Address Output Delay[1]	0.88	4.78
TDRAS	RAS Output Delay[1]	0.79	5.08
TDCAS	CAS Output Delay[1]	0.62	4.80
TDQS	Strobe to Data Setup Time	-1.8	
TDQH	Strobe to Data Hold Time	2.42	
TZOn	DQ/DQS Turn On Time from CLK/$\overline{CLK}$	TBD	
TZOff	DQ/DQS Turn Off Time from CLK/$\overline{CLK}$		TBD

1. To determine delay from NP4GS3 internal clock, add 7.28 ns to Minimum Delay and 11.34 ns to Maximum Delay.
2. Blank space indicates no limit.
3. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 13: PMM Interface Pins

Signal	Description	Type
DMU_A(30:0)	Define the first of the four PMM interfaces and can be configured for TBI, SMII, GMII, or POS. See *Table 14: PMM Interface Pin Multiplexing* on page 44 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
DMU_B(30:0)	Define the second of the four PMM interfaces and can be configured for TBI, SMII, GMII, or POS. See *Table 14: PMM Interface Pin Multiplexing* on page 44 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
DMU_C(30:0)	Define the third of the four PMM interfaces and can be configured for TBI, SMII, GMII, or POS. See *Table 14: PMM Interface Pin Multiplexing* on page 44 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
DMU_D(30:0)	Define the fourth of the four PMM interfaces and can be configured for TBI, SMII, GMII, Debug, or POS. See *Table 14: PMM Interface Pin Multiplexing* on page 44 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_LByte(1:0)	Receive last byte position (valid for 32-bit POS only) provides the position of the last byte within the final word of the packet transfer. This signal is valid only when Rx_EOF is high.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 14: PMM Interface Pin Multiplexing

Pin(s)	Pin Mode		Interface Type				
	DMU_A, DMU_B, DMU_C	DMU_D	GMII	TBI	SMII	Debug (DMU_D only)	8-Bit POS
30	O	O					RxAddr(1) O
29	O	O					RxAddr(0) O
28	O	O					TxAddr(1) O
27	O	O					TxAddr(0) O
26	O	O					TxSOF O
25	O	O	Tx_Valid_Byte O				TxEOF O
(24:17)	O	I/O	Tx_Data(7:0) O	Tx_Data(0:7) O	Tx_Data(9:2) O	Debug(23:16) I/O	TxData(7:0) O
(16:9)	I	I/O	Rx_Data(7:0) I	Rx_Data(0:7) I	Rx_Data(9:2) I	Debug(15:8) I/O	RxData(7:0) I
8	O	O	Tx_Clk 8 ns	Tx_Clk 8 ns	—	—	—
7	O	I/O	Tx_En O	Tx_Data(8) O	Tx_Data(1) O	Debug(7) I/O	TxEn O
6	I/O	I/O	Tx_Er O	Tx_Data(9) O	Tx_Data(0) O	Debug(6) I/O	TxPFA I
5	I	I/O	Rx_Valid_Byte I	Rx_Data(8) I	Rx_Data(1) I	Debug(5) I/O	RxPFA I
4	I	I/O	Tx_Byte_Credit I	Rx_Data(9) I	Rx_Data(0) I	Debug(4) I/O	RxVal I
3	I	I/O	Rx_Clk I 8 ns	Rx_Clk1 I 16 ns	Clk I 8 ns	Debug(3) O	Clk I 10 ns
2	I/O	I/O	Rx_DV I	Rx_Clk0 I 16 ns	Sync O	Debug(2) I/O	RxEOF I
1	I/O	I/O	Rx_Er I	Sig_Det I	Sync2 O	Debug(1) I/O	RxErr I
0	I/O	I/O	$\overline{\text{CPDetect}}$ (0 = CPF) - Input	$\overline{\text{CPDetect}}$ (0 = CPF) - Input Activity - Output	$\overline{\text{CPDetect}}$ (0 = CPF) - Input	Debug(0) I/O	RxEnb O

Table 15: Parallel Data Bit to 8B/10B Position Mapping (TBI Interface)

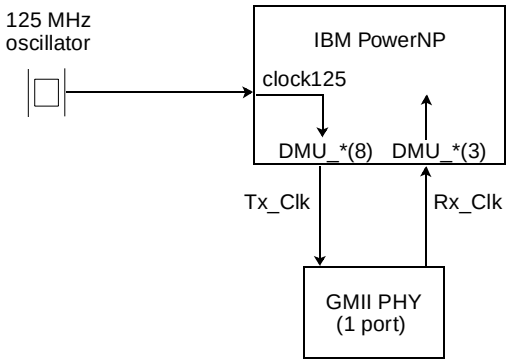
Parallel Data Bit	0	1	2	3	4	5	6	7	8	9
8B/10B Bit Position	a	b	c	d	e	f	g	h	i	j

Table 16: PMM Interface Pins POS32 Mode

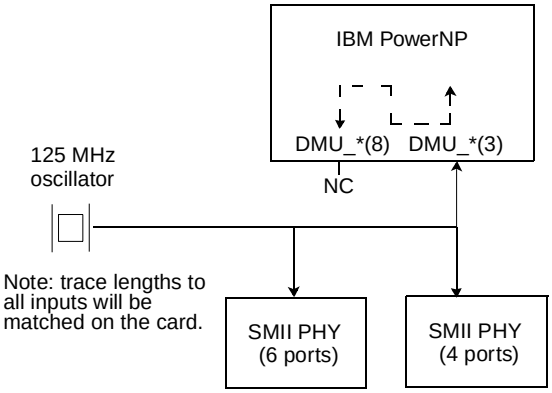
Pin(s)	DMU_A	DMU_B	DMU_C	DMU_D
28	TxPADL(1) O			
27	TxPADL(0) O			
26	TxSOF O			
25	TxEOF O			
(24:17)	TxData(31:24) O	TxData(23:16) O	TxData(15:8) O	TxData(7:0) O
(16:9)	RxData(31:24) I	RxData(23:16) I	RxData(15:8) I	RxData(7:0) I
8	—			
7	TxEn O			
6	TxPFA I			
5	RxPFA I			
4	RxVal I			
3	Clk I 10 ns	Clk I 10 ns	Clk I 10 ns	Clk I 10 ns
2	RxEOF I			
1	RxErr I			
0	RxEnb O			
Single Pins (not associated with a DMU)				
Pin	Rx_LByte			
1	Rx_LByte(1) I			
0	Rx_LByte(0) I			

Figure 9: NP4GS3 DMU Bus Clock Connections

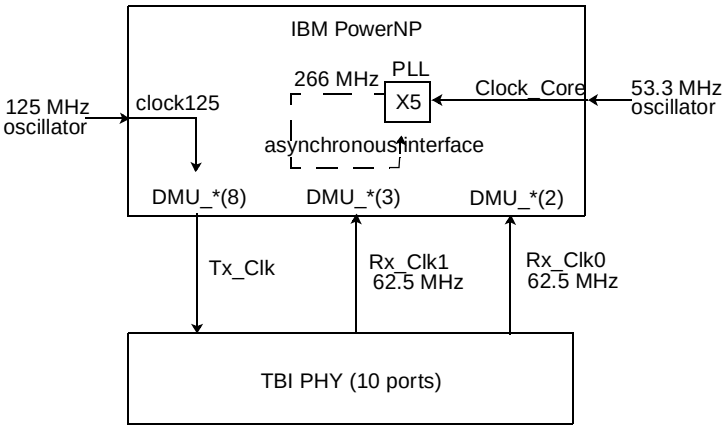
GMII Interface



SMII Interface



TBI Interface



Notes: Each figure above illustrates a single DMU bus and applies to any of the four DMU busses.
The “DMU_*” labels represent any of the four DMU busses (DMU_A, DMU_B, DMU_C, or DMU_D).

Figure 10: NP4GS3 DMU Bus Clock Connections (POS Overview)

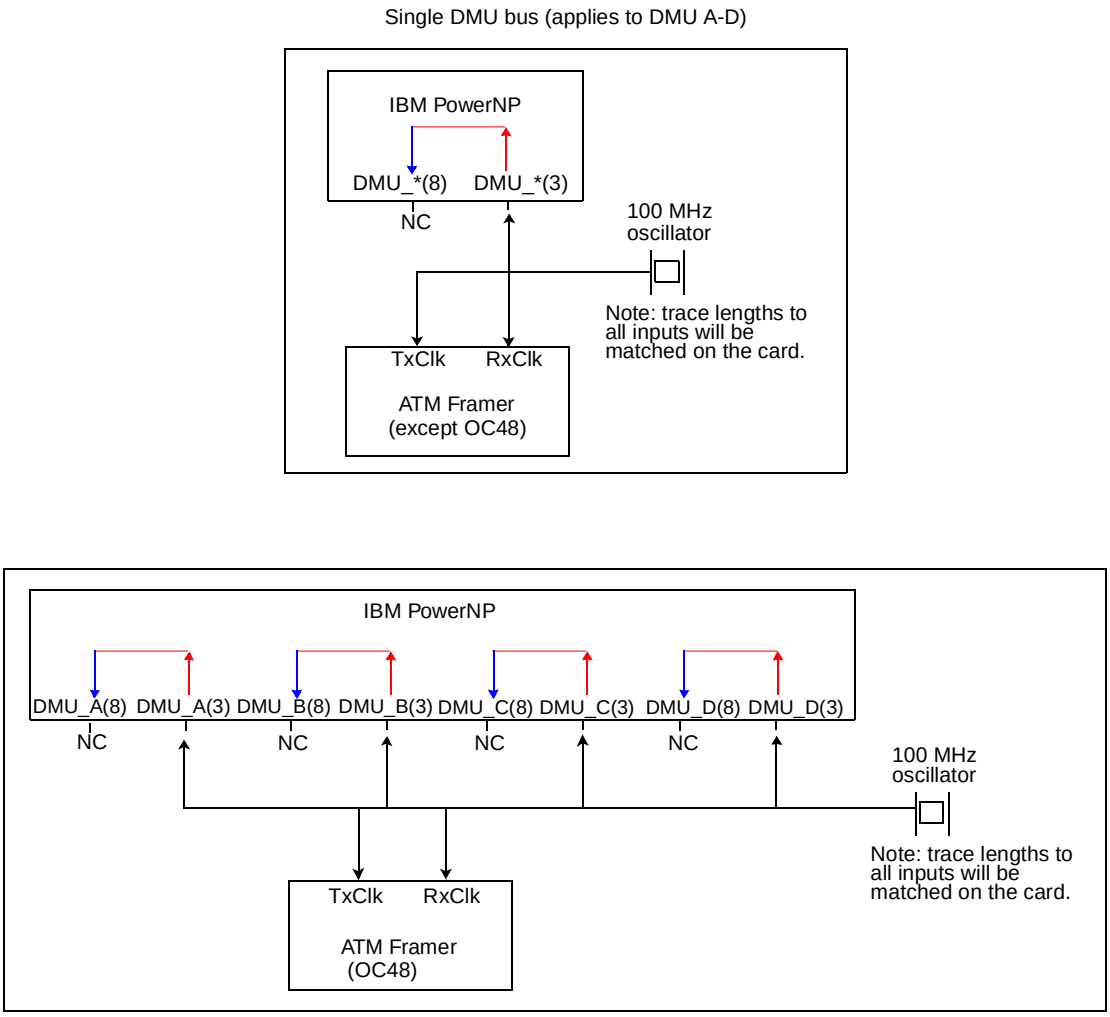


Table 17: PMM Interface Signals: GMII Mode (Refer to notes)

Signal	Description	Type
Tx_Data(7:0)	Transmit Data. Data bus to the PHY, synchronous to Tx_Clk.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Data(7:0)	Received Data. Data bus from the PHY, synchronous to Rx_Clk.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_En	Transmit data Enabled to the PHY, synchronous to Tx_Clk. 0 End of frame transmission 1 Active frame transmission	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_Er	Transmit Error, synchronous to the Tx_Clk. 0 No error detected 1 Informs the PHY that MAC detected an error	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Valid_Byte	Receive valid data, synchronous to the Rx_Clk. 0 Data invalid 1 Byte of data (from the PHY) on Rx_Data is valid. For a standard GMII connection, this signal can be tied to '1' on the card.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_Byte_Credit	Transmit next data value, asynchronous. 0 Do not send next data byte 1 Asserted. PHY indicates that the next Tx_Data value may be sent. For a standard GMII connection, this signal can be tied to '1' on the card.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_Valid_Byte	Transmit valid data, synchronous to Tx_Clock 0 Data invalid 1 Byte of data (from the Network Processor) on Tx_Data is valid.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Clk	125 MHz Receive Medium clock generated by the PHY.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_DV	Receive Data Valid (from the PHY), synchronous to Rx_Clk. 0 End of frame transmission. 1 Active frame transmission.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Er	Receive Error, synchronous to Rx_Clk. 0 No error detected 1 Informs the MAC that PHY detected an error	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_Clk	125 MHz transmit clock to the PHY. During operation, the network processor drives this signal to indicate that a transmit is in progress for this interface.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{CPDetect}}$	The Control Point card drives this signal active low to indicate its presence. When a non-Control Point card is plugged in, or this device pin is not connected, this signal should be pulled to a '1' on the card.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V

1. The NP4GS3 supports GMII in Full-Duplex Mode only.
2. See *Table 14: PMM Interface Pin Multiplexing* on page 44 for pin directions (I/O) and definitions.

Figure 11: GMII Timing Diagram

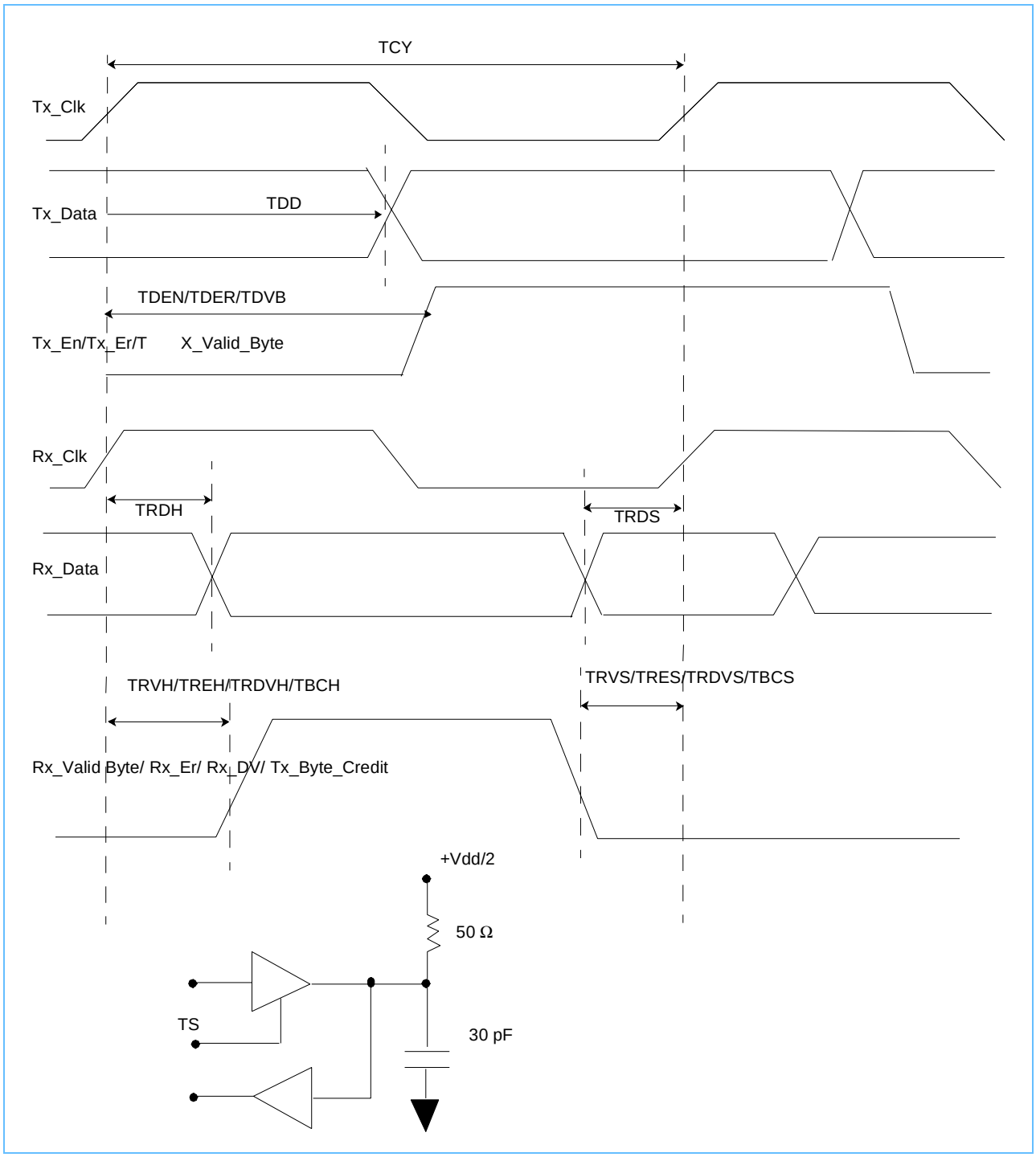


Table 18: GMII Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	GMII Cycle Time	8	
TDD	Tx_Data Output Delay[1]	3.67	4.71
TDER	Tx_Er Output Delay[3]	3.82	4.65
TDVB	Tx_Valid_Byte Output Delay[2]	3.77	4.77
TDEN	Tx_En Output Delay	3.78	4.67
TRDS	Rx_Data Setup Time	1.78	
TRDH	Rx_Data Hold Time	0	
TRVS	Rx_Valid_Byte Setup Time	1.78	
TRVH	Rx_Valid_Byte Hold Time	0	
TRES	Rx_Er Setup Time	1.86	
TREH	Rx_Er Hold Time	0	
TRDVS	Rx_DV Setup Time	1.56	
TRDVH	Rx_DV Hold Time	0	
TBCS	Tx_Byte_Credit Setup Time	1.71	
TBCH	Tx_Byte_Credit Hold Time	0	

1. To determine delay from NP4GS3 internal clock, add -0.58 ns to Minimum Delay and 1.26 ns to Maximum Delay.
2. To determine delay from NP4GS3 internal clock, add -0.73 ns to Minimum Delay and 1.35 ns to Maximum Delay.
3. To determine delay from NP4GS3 internal clock, add 1.49 ns to Minimum Delay and 1.49 ns to Maximum Delay.
4. Blank space indicates no limit.
5. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 19: PMM Interface Pins: TBI Mode

Signal	Description	Type
Tx_Data(9:0)	Transmit data. Data bus to the PHY, synchronous to Tx_Clk.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Data(9:0)	Receive data. Data bus from the PHY, synchronous to Rx_Clk1 and Rx_Clk0. (Data switches at double the frequency of Rx_Clk1 or Rx_Clk0.)	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Clk1	Receive Clock, 62.5 MHz. Rx_Data is valid on the rising edge of this clock.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Clk0	Receive Clock, 62.5 MHz. This signal is 180 degrees out of phase with Rx_Clk1. Rx_Data is valid on the rising edge of this clock.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Note: See *Table 14: PMM Interface Pin Multiplexing* on page 44 table for pin directions (I/O) and definitions.

Table 19: PMM Interface Pins: TBI Mode

Signal	Description	Type
Sig_Det	Signal Detect. Signal asserted by the PHY to indicate that the physical media is valid.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{CPDetect}}$	The Control Point card drives this signal active low to indicate its presence. When a non-Control Point card is plugged in, or this device pin is not connected, this signal should be pulled to a '1' on the card. During operation, this signal is driven by the network processor to indicate the status of the interface. 0 TBI interface is not in the data pass state (link down) 1 TBI interface is in the data pass state (occurs when auto-negotiation is complete, or when idles are detected (if AN is disabled)) pulse TBI interface is in a data pass state and is either receiving or transmitting. The line pulses once per frame transmitted or received at a maximum rate of 8Hz.	Input/Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Tx_Clk	125 MHz clock Transmit clock to the PHY. During operation, the network processor drives this signal to indicate that a transmit or receive is in progress for this interface.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Note: See *Table 14: PMM Interface Pin Multiplexing* on page 44 table for pin directions (I/O) and definitions.

Figure 12: TBI Timing Diagram

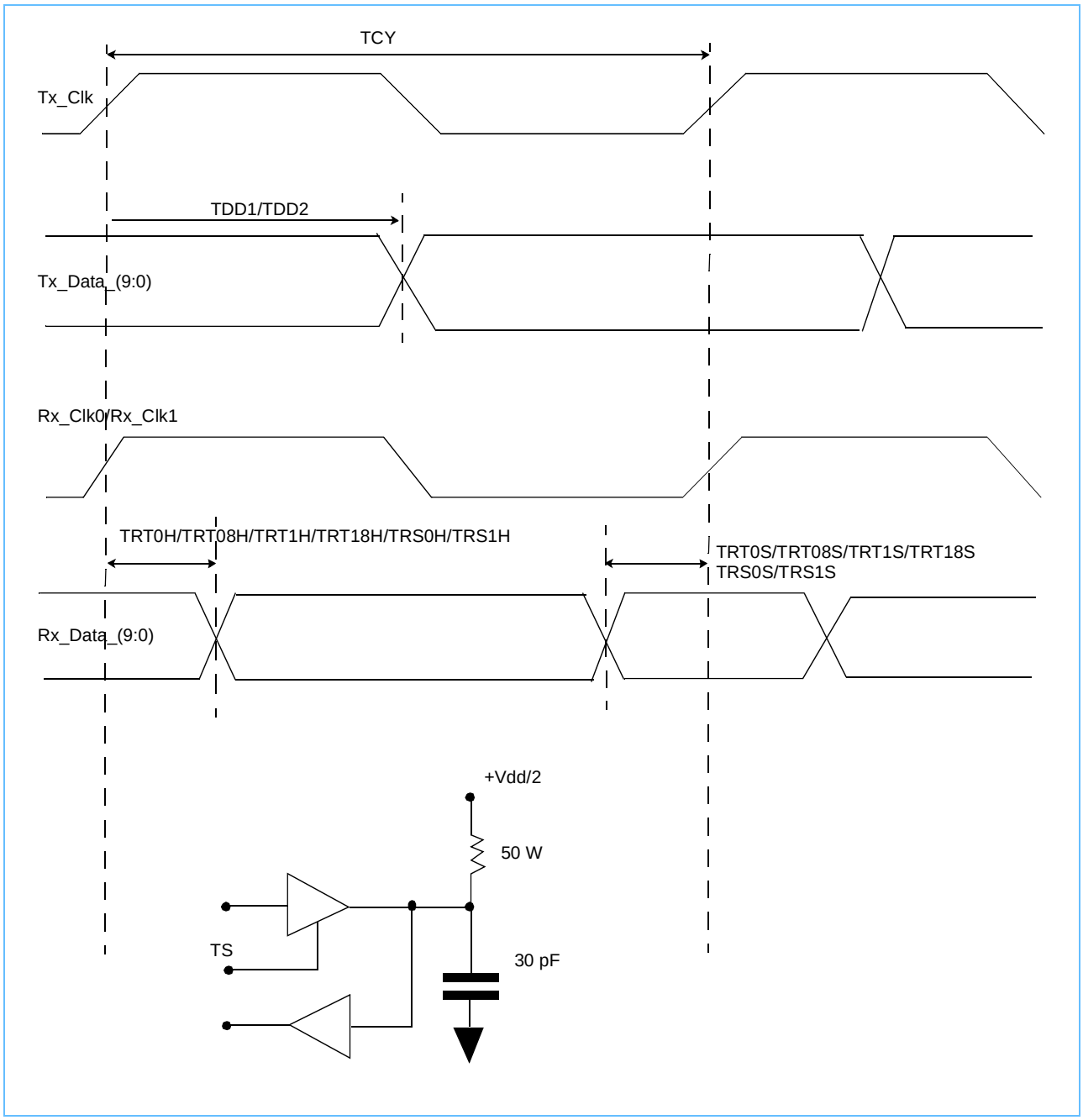


Table 20: TBI Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	TBI Cycle Time	16	
TDD1	Tx_Data_(7:0) Output Delay[1]	3.67	4.71
TDD2	Tx_Data_(9:8) Output Delay[1]	3.67	4.71
TRT0S	Rx_Data_(7:0) Set Up Time Clk0	0.66	
TRT0H	Rx_Data_(7:0) Hold Time Clk0	0.72	
TRT08S	Rx_Data_(9:8) Set Up Time Clk0	0.66	
TRT08H	Rx_Data_(9:8) Hold Up Time Clk0	0.72	
TRS0S	Sig_Det Setup Time Clk0	0.66	
TRS0H	Sig_Det Hold Time Clk0	0.72	
TRT1S	Rx_Data_(7:0) Set Up Time Clk1	0.80	
TRT1H	Rx_Data_(7:0) Hold Time Clk1	0.12	
TRT18S	Rx_Data_(9:8) Set Up Time Clk1	0.80	
TRT18H	Rx_Data_(9:8) Hold Up Time Clk1	0.12	
TRS1S	Sig_Det Setup Time Clk1	0.80	
TRS1H	Sig_Det Hold Time Clk1	0.12	

1. To determine delay from NP4GS3 internal clock, add -0.58 ns to Minimum Delay and 1.26 ns to Maximum Delay.
2. Blank space indicates no limit.
3. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 21: PMM Interface Pins: SMII Mode

Signal	Description	Type
Tx_Data(9:0)	Transmit Data. Data bus to the PHY - contains ten streams of serial transmit data. Each serial stream is connected to a unique port. Synchronous to the common clock (Clk).	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Rx_Data(9:0)	Received Data. Data bus from the PHY - contains ten streams of serial receive data. Each serial stream is connected to a unique port. Synchronous to the common clock (Clk).	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Sync	Asserted for one Tx_Clk cycle once every ten Tx_Clk cycles. Assertion indicates the beginning of a 10-bit segment on both Tx_Data and Rx_Data.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Sync2	Logically identical to Sync and provided for fanout purposes.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{CPDetect}}$	The Control Point card drives this signal active low to indicate its presence. When a non-Control Point card is plugged in, or this device pin is not connected, this signal should be pulled to a '1' on the card.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Note: The transmit and receive clocks the NP4GS3 uses for this mode are derived from the Clock 125 input. See *Figure 9* on page 46. See *Table 14: PMM Interface Pin Multiplexing* on page 44 table for pin directions (I/O) and definitions.

Figure 13: SMII Timing Diagram

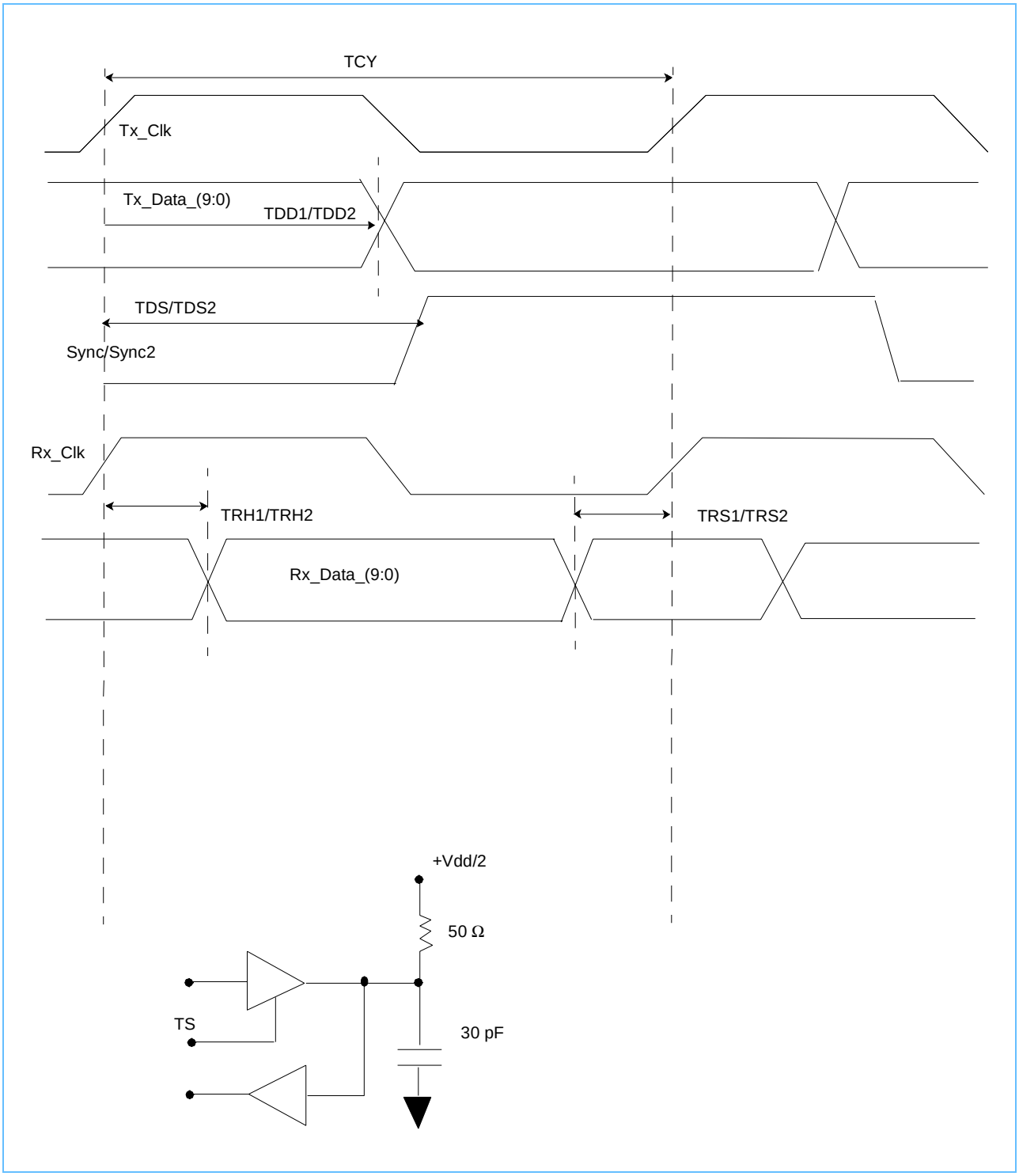


Table 22: SMII Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	SMII Cycle Time	8	
TDD1	Tx_Data_(9:2) Output Delay[1]	2.07	4.29
TDD2	Tx_Data_(1:0) Output Delay[1]	2.13	4.31
TDS	Sync Output Delay[1]	2.12	4.23
TDS2	Sync2 Output Delay[1]	2.10	4.48
TRS1	Rx_Data_(9:2) Set Up Time	0.37	
TRH1	Rx_Data_(9:2) Hold Time	1.22	
TRS2	Rx_Data_(1:0) Set Up Time	0.19	
TRH2	Rx_Data_(1:0) Hold Time	1.19	

1. To determine delay from NP4GS3 internal clock, add 0 ns to Minimum Delay and 0 ns to Maximum Delay.
2. Blank space indicates no limit.
3. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 23: POS Signals

Signal	Description	Type
RxAddr(1:0)	Receive address bus selects a particular port in the framer for a data transfer. Valid on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
RxData (7:0) 8-bit mode (31:0) 32-bit mode	Receive POS data bus carries the frame word that is read from the Framer's FIFO. RxData transports the frame data in an 8-bit format. RxData[7:0] and [31:0] are updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
Clk	POS clock provides timing for the POS Framer interface. Clk must cycle at a 100 MHz or lower instantaneous rate.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{RxEnb}}$	Receive read enable controls read access from the Framer's receive interface. The framer's addressed FIFO is selected on the falling edge of RxEnb. Generated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
RxEOF	Receive end-of-frame marks the last word of a frame in RxData. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
RxErr	Receive packet error indicates that the received packet contains an error and must be discarded. Only asserted on the last word of a packet (when RxEOF is also asserted). Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
RxVal	Receive valid data output indicates the receive signals RxData, RxSOF, RxEOF, RxErr, and RxPADL are valid from the framer. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
RxPADL(1:0)	Receive padding length indicates the number of padding bytes included in the last word of the packet transferred in RxData. Only used when the network processor is configured in 32-bit POS mode. Updated on the rising edge of Clk. RxPADL(1:0) (32-bit mode) 00 packet ends on RxData(7:0) (RxData = DDDD) 01 packet ends on RxData(15:8) (RxData = DDDP) 10 packet ends on RxData(23:16) (RxData = DDPP) 11 packet ends on RxData(31:24) (RxData = DPPP)	5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 23: POS Signals (Continued)

Signal	Description	Type
RxPFA	Receive polled frame-available input indicates that the framers polled receive FIFO contains data. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxData (7:0) 8-bit mode (31:0) 32-bit mode	Transmit UTOPIA data bus carries the frame word that is written to the framer's transmit FIFO. Considered valid and written to a framer's transmit FIFO only when the transmit interface is selected by using TxEnb. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxEn	Transmit write enable controls write access to the transmit interface. A framer port is selected on the falling edge of TxEnb. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxAddr(1:0)	Transmit address bus uses TxEnb to select a particular FIFO within the framer for a data transfer. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxSOF	Transmit start-of-frame marks the first word of a frame in TxData. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxEOF	Transmit end-of-frame marks the last word of a frame in TxData. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxPADL (1:0)	Transmit padding length indicates the number of padding bytes included in the last word of the packet transferred in TxData. Sampled on the rising edge of Clk. When configured in 32-bit mode the last word may contain zero, one, two or three padding bytes and only TxPADL[1:0] is used. In 8-bit mode TxPADL[1:0] is not used. TxPADL[1:0] (32-bit mode) 00 packet ends on TxData[7:0] (TxData = DDDD) 01 packet ends on TxData[15:8] (TxData = DDDP) 10 packet ends on TxData[23:16] (TxData = DDPP) 11 packet ends on TxData[31:24] (TxData = DPPP)	5.0 V-tolerant 3.3 V LVTTL 3.3 V
TxPFA	Transmit polled frame-available output - indicates that the polled framer's transmit FIFO has free available space and the NP4GS3 can write data into the framer's FIFO. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL 3.3 V

Figure 14: POS Timing Diagram

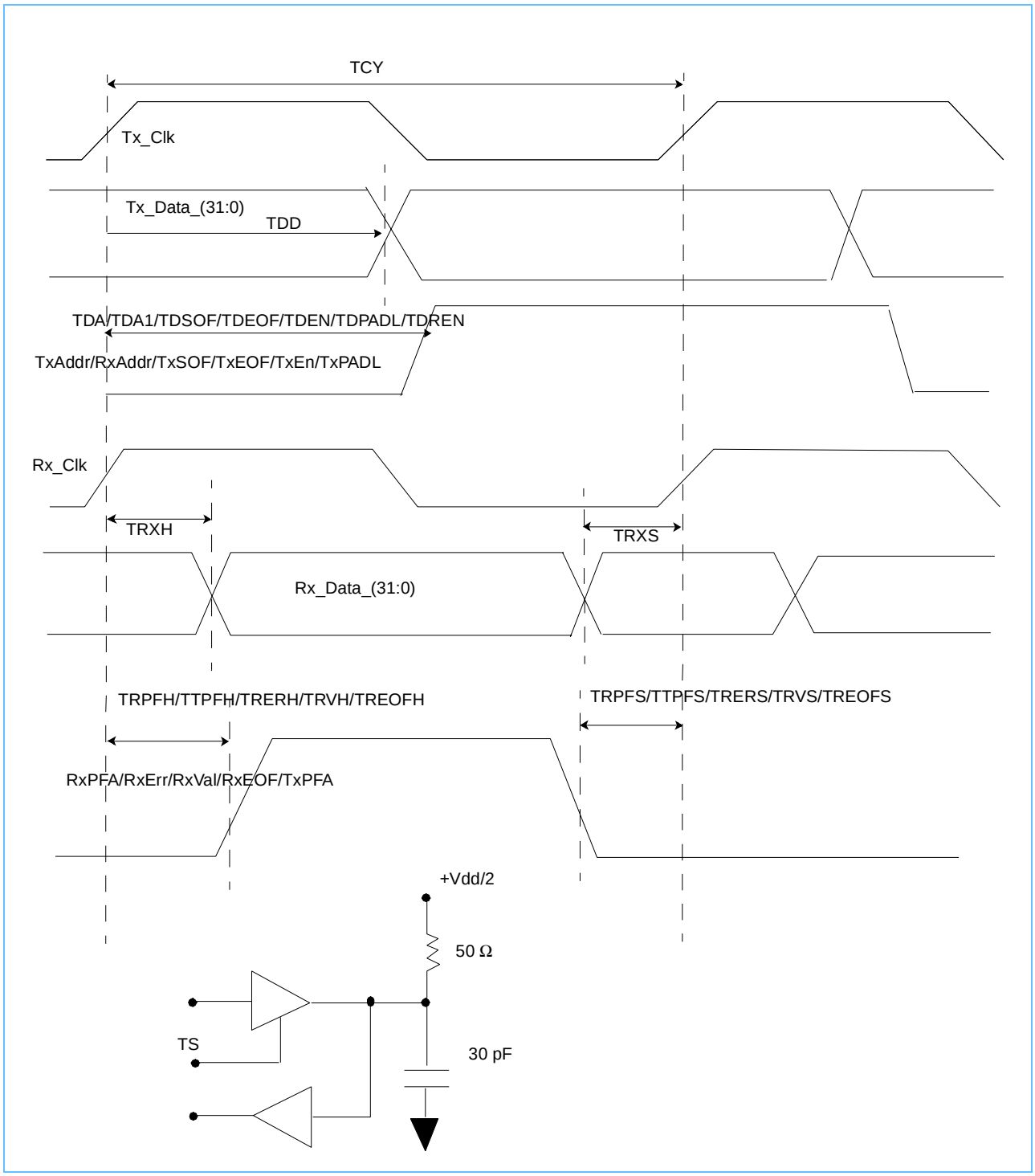


Table 24: POS Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	POS Cycle Time	10	
TDD	Tx_Data_(31:0) Output Delay[1]	2.07	4.29
TDA	Tx_Addr_(1:0) Output Delay[1]	2.12	4.34
TDA1	Rx_Addr_(1:0) Output Delay[1]	1.87	3.9
TDSOF	TxSOF Output Delay[1]	2.13	4.22
TDEOF	TxEOF Output Delay[1]	2.15	4.40
TDEN	TxEn Output Delay[1]	1.79	4.31
TDPADL	TxPADL_(1:0) Output Delay[1]	2.12	4.34
TDREN	RxEnb Output Delay[1]	1.79	3.85
TRXS	Rx_Data_(31:0) Set Up Time	1.3	
TRXH	Rx_Data_(31:0) Hold Time	0.25	
TRVS	RxVal Set Up Time	1.09	
TRVH	RxVal Hold Time	0.35	
TRERS	RxErr Set Up Time	1.2	
TRERH	RxErr Hold Time	0	
TREOFS	RxEOF Set Up Time	1.00	
TREOFH	RxEOF Hold Time	0	
TRPFS	RxPFA Setup Time	1.19	
TRPFH	RxPFA Hold Time	0	
TTPFS	TxPFA Setup Time	1.09	
TTPFH	TxPFA Hold Time	0	

1. To determine delay from NP4GS3 internal clock, add 0 ns to Minimum Delay and 0 ns to Maximum Delay.
2. Blank space indicates no limit.
3. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.
4. Tx_Clk is not an external pin. It is an internal clock derived from the 100 MHz clock

Table 25: PCI Interface Pins

Signal	Description	Type
PCI_Clk	PCI Clock Signal (See PCI_Speed field below)	Input PCI 3.3 V
PCI_AD(31:0)	PCI Multiplexed Address and Data Signals	Input/Output PCI 3.3 V
$\overline{\text{PCI\_CBE}}$(3:0)	PCI Command/Byte Enable Signals	Input/Output PCI 3.3 V
PCI_Frame	PCI Frame signal	Input/Output PCI 3.3 V
PCI_IRdy	PCI Initiator (Master) Ready Signal	Input/Output PCI 3.3 V
PCI_TRdy	PCI Target (Slave) Ready Signal	Input/Output PCI 3.3 V
PCI_DevSel	PCI Device Select Signal	Input/Output PCI 3.3 V
$\overline{\text{PCI\_Stop}}$	PCI Stop Signal	Input/Output PCI 3.3 V
$\overline{\text{PCI\_Request}}$	PCI Bus Request Signal	Output PCI 3.3 V
$\overline{\text{PCI\_Grant}}$	PCI Bus Grant Signal	Input PCI 3.3 V
PCI_IDSel	PCI Initialization Device Select Signal	Input PCI 3.3 V
$\overline{\text{PCI\_PErr}}$	PCI Parity Error Signal	Input/Output PCI 3.3 V
$\overline{\text{PCI\_SErr}}$	PCI System Error Signal	Input/Output PCI 3.3 V
PCI_IntA	PCI Level Sensitive Interrupt	Output PCI 3.3 V
PCI_Par	PCI Parity Signal. Covers all the data/address and the four command/BE signals.	Input/Output PCI 3.3 V
PCI_Speed	PCI Speed. Frequency of attached PCI bus. 0 66 MHz 1 33 MHz	Input 3.3 V-tolerant 2.5 V 2.5 V
PCI_Bus_NM_Int	External non-maskable interrupt - the active polarity of the interrupt is programmable by the PowerPC.	Input PCI 3.3 V

Note: PCI I/O are all configured for multi-point operation.

Table 25: PCI Interface Pins (Continued)

Signal	Description	Type
PCI_Bus_M_Int	External maskable interrupt - the active polarity of the interrupt is programmable by the PowerPC.	Input PCI 3.3 V

Note: PCI I/O are all configured for multi-point operation.

Figure 15: PCI Timing Diagram

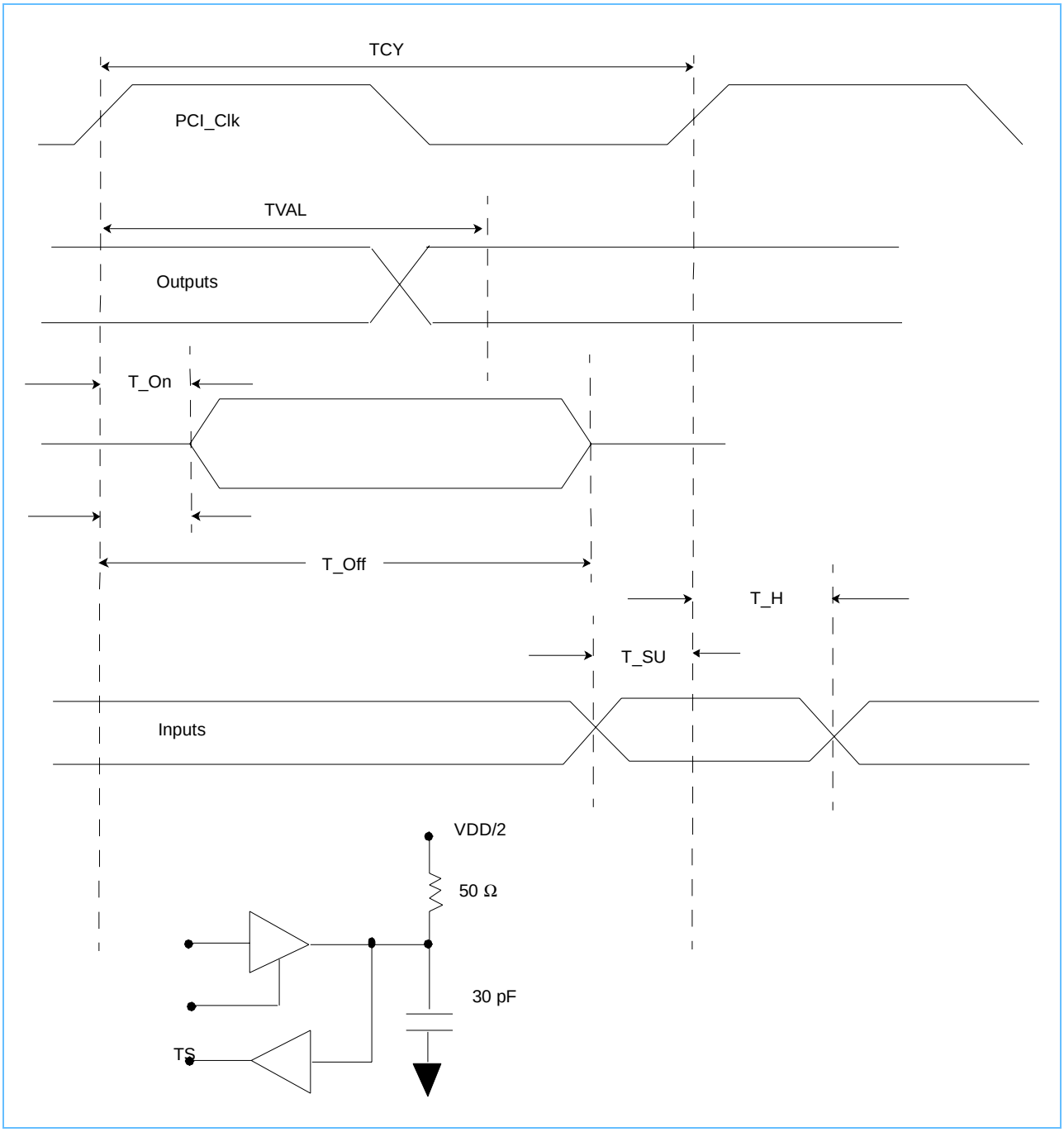


Table 26: PCI Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	PCI Cycle Time	15	
TVAL	Worst Case Output Delay	2.79	5.7
T_On	PCI Bus Turn on Output Delay	2	
T_Off	PCI Bus Turn off Output Delay		14
T_SU	Input Set Up Time	2.12	
T_H	Input Hold Time	0	

1. Blank space indicates no limit.
2. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 27: Management Bus Pins

Signal	Description	Type
MG_Data	Serial Data. Supports Address/Control/Data protocol.	Input/Output 3.3 V-tolerant 2.5 V 2.5 V
MG_Clk	33.33 MHz clock	Output 3.3 V-tolerant 2.5 V 2.5 V
MG_nIntr	Rising-edge sensitive interrupt input	Input 3.3 V-tolerant 2.5 V 2.5 V

Figure 16: Management Bus Timing Diagram

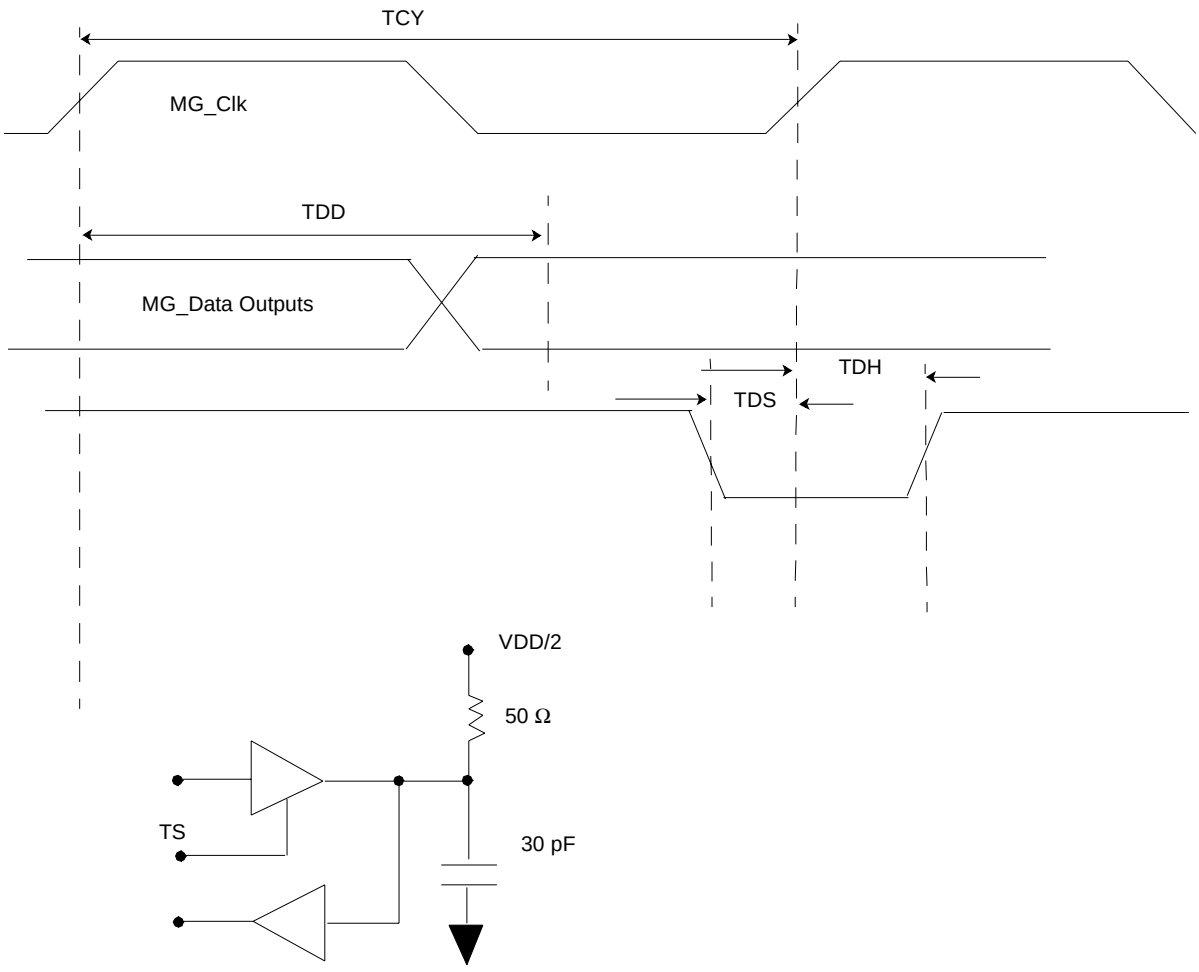


Table 28: Management Bus Timing Diagram Legend

Symbol	Symbol Description	Minimum Delay (ns)	Maximum Delay (ns)
TCY	SPM Cycle Time	30	
TDD	Worst Case Data Output Delay[1]	0.5	0.82
TDS	Data Setup Time	1.6	
TDH	Data Hold Time	0	

1. To determine delay from NP4GS3 internal clock, add 6.59 ns to Minimum Delay and 7.09 ns to Maximum Delay.
2. MG_nIntr is an asynchronous input and is not timed.
3. Blank space indicates no limit.
4. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 29: Miscellaneous Pins

Signal	Description	Type
Switch_Clock_A	The positive pin of an input differential pair. 50.6875 to 62.5 MHz. Generates Packet Routing Switch clock domains. Required to have cycle-to-cycle jitter ≤ ±150ps. Duty Cycle tolerance must be ≤ ±10%.	Input LVDS 1.5 V
$\overline{\text{Switch\_Clock\_A}}$	The negative pin of an input differential pair. 50.6875 to 62.5 MHz.	Input LVDS 1.5 V
Switch_Clock_B	The positive pin of an input differential pair. 50.6875 to 62.5 MHz. Generates Packet Routing Switch clock domains. Required to have cycle-to-cycle jitter ≤ ±150ps. Duty Cycle tolerance must be ≤ ±10%.	Input LVDS 1.5 V
$\overline{\text{Switch\_Clock\_B}}$	The negative pin of an input differential pair. 50.6875 to 62.5 MHz.	Input LVDS 1.5 V
Switch_BNA	Selects which of the two DASL ports (A or B) carries network traffic. 0 Port A carries the network traffic 1 Port B carries the network traffic	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Core_Clock	53.33 MHz oscillator - generates 266 /133 clock domains. Required to have cycle-to-cycle jitter ≤ ±150ps. Duty Cycle tolerance must be ≤ ±5%.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Clock125	125 MHz oscillator. Required to have cycle-to-cycle jitter ≤ ±60ps. Duty Cycle tolerance must be ≤ ± 5%.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{Blade\_Reset}}$	Reset NP4GS3 - signal must be driven active low for a minimum of 1µs to ensure a proper reset of the NP4GS3. All input clocks (Switch_Clock_A, $\overline{\text{Switch\_Clock\_A}}$, Switch_Clock_B, $\overline{\text{Switch\_Clock\_B}}$, Core_Clock, Clock125, and PCI_Clk) must be running prior to the activation of this signal.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
$\overline{\text{Operational}}$	NP4GS3 operational - pin is driven active low when both the NP4GS3 Ingress and Egress Macros have completed their initialization. It remains active until a subsequent $\overline{\text{Blade\_Reset}}$ is issued.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V
Testmode(1:0)	00 Functional Mode, including concurrent use of the JTAG interface for RISCWatch or CABWatch operations. 01 Debug Mode - Debug mode must be indicated by the Testmode I/O for the debug bus (DMU_D) output to be gated from the probe. 10 JTAG Test Mode 11 LSSD Test Mode	Input CMOS 1.8 V
$\overline{\text{JTAG\_TRst}}$	JTAG test reset. For normal functional operation, this pin must be connected to the same card source that is connected to the $\overline{\text{Blade\_Reset}}$ input. When the JTAG interface is used for JTAG test functions, this pin is controlled by the JTAG interface logic on the card.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
JTAG_TMS	JTAG test mode select. For normal functional operation, this pin should be tied either low or high.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
JTAG_TDO	JTAG test data out. For normal functional operation, this pin should be tied either low or high.	Output 5.0 V-tolerant 3.3 V LVTTL 3.3 V

Table 29: Miscellaneous Pins (Continued)

Signal	Description	Type
JTAG_TDI	JTAG test data in. For normal functional operation, this pin should be tied either low or high.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
JTAG_TCk	JTAG test clock. For normal functional operation, this pin should be tied either low or high.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3 V
PLLA_V_{DD} PLLB_V_{DD} PLLC_V_{DD}	These pins serve as the +1.8 Volt supply for a critical noise sensitive portion of the Phased Locked Loop (PLL) circuits. One pin serves as the analog V_{DD} for each PLL circuit. To prevent noise on these pins from introducing phase jitter in the PLL outputs, place filters at the board level to isolate these pins from the noisy digital V_{DD} pins. Place separate filters on each analog V_{DD} pin to prevent noise from one PLL being introduced into another. See section *2.1.1 PLL Filter Circuit on page 65* for filter circuit details.	Input PLL_V_{DD} 1.8 V
PLLA_GND PLLB_GND PLLC_GND	These pins serve as the ground connection for the critical noise portion of the Phase Lock Loop (PLL). One pin serves as the analog GND for each PLL circuit. Each should be connected to the digital ground plane at the V_{DDA} node of the PLL filter capacitor shown in *Figure 17: PLL Filter Circuit Diagram* on page 66.	Input PLL_GND 0.0 V
Thermal_In	Input pad of the thermal monitor (resistor). See *2.1.2 Thermal I/O Usage* on page 66 for details on thermal monitor usage.	Thermal
Thermal_Out	Output pad of the thermal monitor (resistor).	Thermal
VRef1(2), VRef2(8,7,6)	Voltage reference for SSTL2 I/Os for D1, D2, and D3 (approximately four pins per side of the device that contains SSTL2 I/O).	Input VRef 1.25 V
VRef1(1), VRef2(5,4,3)	Voltage reference for SSTL2 I/Os for D4 and D6 (approximately four pins per side of the device that contains SSTL2 I/O).	Input VRef 1.25 V
VRef1(0), VRef2(2,1,0)	Voltage reference for SSTL2 I/Os for DS0 and DS1 (approximately four pins per side of the device that contains SSTL2 I/O).	Input VRef 1.25 V
Boot_Picocode	Determines location of network processor picocode load location. 0 Load from SPM 1 Load from external source (typically Power PC or PCI bus)	Input 3.3 V-tolerant 2.5 V 2.5 V
Boot_PPC	Determines location of Power PC code start location. 0 Start from D6 1 Start from PCI	Input 3.3 V-tolerant 2.5 V 2.5 V
Spare_Tst_Rcvr(9:0)	Unused signals needed for Manufacturing Test. Spare_Tst_Rcvr (9:5,1) should be tied to 0 on the card. Spare_Tst_Rcvr (4:2,0) should be tied to 1 on the card.	Input CMOS 1.8 V
$\overline{\text{C405\_Debug\_Halt}}$	This signal, when asserted low, forces the embedded PowerPC 405 processor to stop processing all instructions. For normal functional operation, this signal should be tied inactive high.	Input 5.0 V-tolerant 3.3 V LVTTL 3.3V

Table 30: Signals Requiring Pull-Up or Pull-Down

Signal Name	Function	Value
Signals requiring a DC connection that is the same value for all applications		
Testmode(1:0)		Pull-down
JTAG_TDI		Pull-up
JTAG_TMS		Pull-up
JTAG_TCk		Pull-up
$\overline{\text{C405\_Debug\_Halt}}$		Pull-up
Spare_Tst_Rcvr (9:5, 1)		Pull-down
Spare_Tst_Rcvr (4:2, 0)		Pull-up
Signals requiring a DC connection that varies across different applications		
Multicast_Grant_A, Multicast_Grant_B		Pull-up if no system device drives this signal
RES_Data		Pull-down if no other system device drives this signal
PCI_Speed		Choose up or down based on system PCI bus speed
MG_nIntr		Pull-down if no system device drives this signal
Boot_Picocode		Choose up or down based on Picocode load location
Boot_PPC		Choose up or down based on PPC code load location
Switch_BNA		Pull-up if no system device drives this signal
Signals which have an AC connection, but also require pull-up or pull-down		
$\overline{\text{Operational}}$		Pull-up
DMU_A(0), DMU_B(0), DMU_C(0), DMU_D(0)	$\overline{\text{CPDetect}}$	If control point blade then pull-down, otherwise pull-up
DMU_A(30:29), DMU_B(30:29), DMU_C(30:29), DMU_D(30:29)	DMU in 8-bit POS mode. RxAddr (1:0)	Pull-down
DMU_A(4), DMU_B(4), DMU_C(4), DMU_D(4)	DMU in any POS mode. RxVal	Pull-down
D3_DQS(1:0), D2_DQS(1:0), D1_DQS(1:0)		Pull-down
D0_DQS(3:0), D4_DQS(3:0), D6_DQS(3:0)		Pull-down
DS0_DQS(3:0), DS1_DQS(3:0)		Pull-down

2.1.1 PLL Filter Circuit

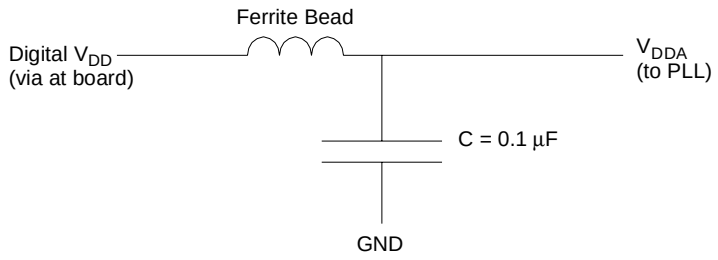
V_{DDA} is the voltage supply pin to the analog circuits in the PLL. Noise on V_{DDA} causes phase jitter at the output of the PLL. V_{DDA} is brought to a package pin to isolate it from the noisy internal digital V_{DD} signal. If little noise is expected at the board level, then V_{DDA} can be connected directly to the digital V_{DD} plane. In most circumstances, however, it is prudent to place a filter circuit on the V_{DDA} as shown below.

Note: All wire lengths should be kept as short as possible to minimize coupling from other signals.

The impedance of the ferrite bead should be much greater than that of the capacitor at frequencies where noise is expected. Many applications have found that a resistor does a better job of reducing jitter than a ferrite bead does. The resistor should be kept to a value lower than 2Ω. Experimentation is the best way to determine the optimal filter design for a specific application.

Note: One filter circuit may be used for PLLA and PLLB, and a second filter circuit should be used for PLLC.

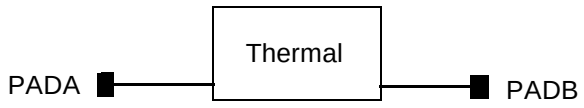
Figure 17: PLL Filter Circuit Diagram



2.1.2 Thermal I/O Usage

The thermal monitor consists of a resistor connected between pins PADA and PADB. At 25 °C this resistance is estimated at 1290 + 350 ohms. The published temperature coefficient of the resistance for this technology is 0.33% per °C. To determine the actual temperature coefficient, see *Measurement Calibration* on page 66.

Figure 18: Thermal Monitor



Note: There is an electrostatic discharge (ESD) diode at PADA and PADB.

2.1.2.1 Temperature Calculation

The chip temperature can be calculated from

$$T_{chip} = \frac{1}{t_c (R_{measured} - R_{calibrated})} + T_{calibrated}\ °C$$

where:

$R_{measured}$ = resistance measured between PADA and PADB at test temperature.

$R_{calibrated}$ = resistance measured between PADA and PADB (V r /I dc) at known temperature.

$T_{calibrated}$ = known temperature used to measure R calibrated.

2.1.2.2 Measurement Calibration

To use this thermal monitor accurately, it must first be calibrated. To calibrate, measure the voltage drop at

two different known temperatures at the package while the chip is dissipating little (less than 100 mW) or no power. Apply I_{dc} and wait for a fixed time t_m, where t_m = approx. 1 ms. Keep t_m short to minimize heating effects on the thermal monitor resistance. Then measure V_r. Next, turn off I_{dc} and change the package temperature. Reapply I_{dc}, wait t_m again and measure V_r.

The temperature coefficient is,

$$Tc = \left[\frac{\Delta Vr}{Idc \times \Delta T}\right]\frac{\Omega}{°C} \quad ,$$

where:

ΔT = temperature change, °C

ΔV_r = voltage drop, V

I_{dc} = applied current, A

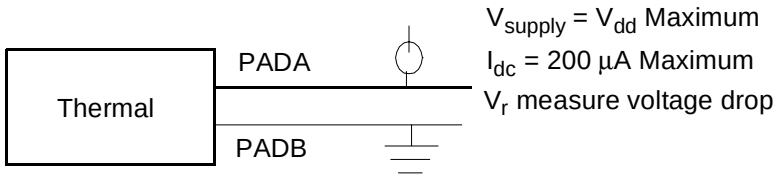
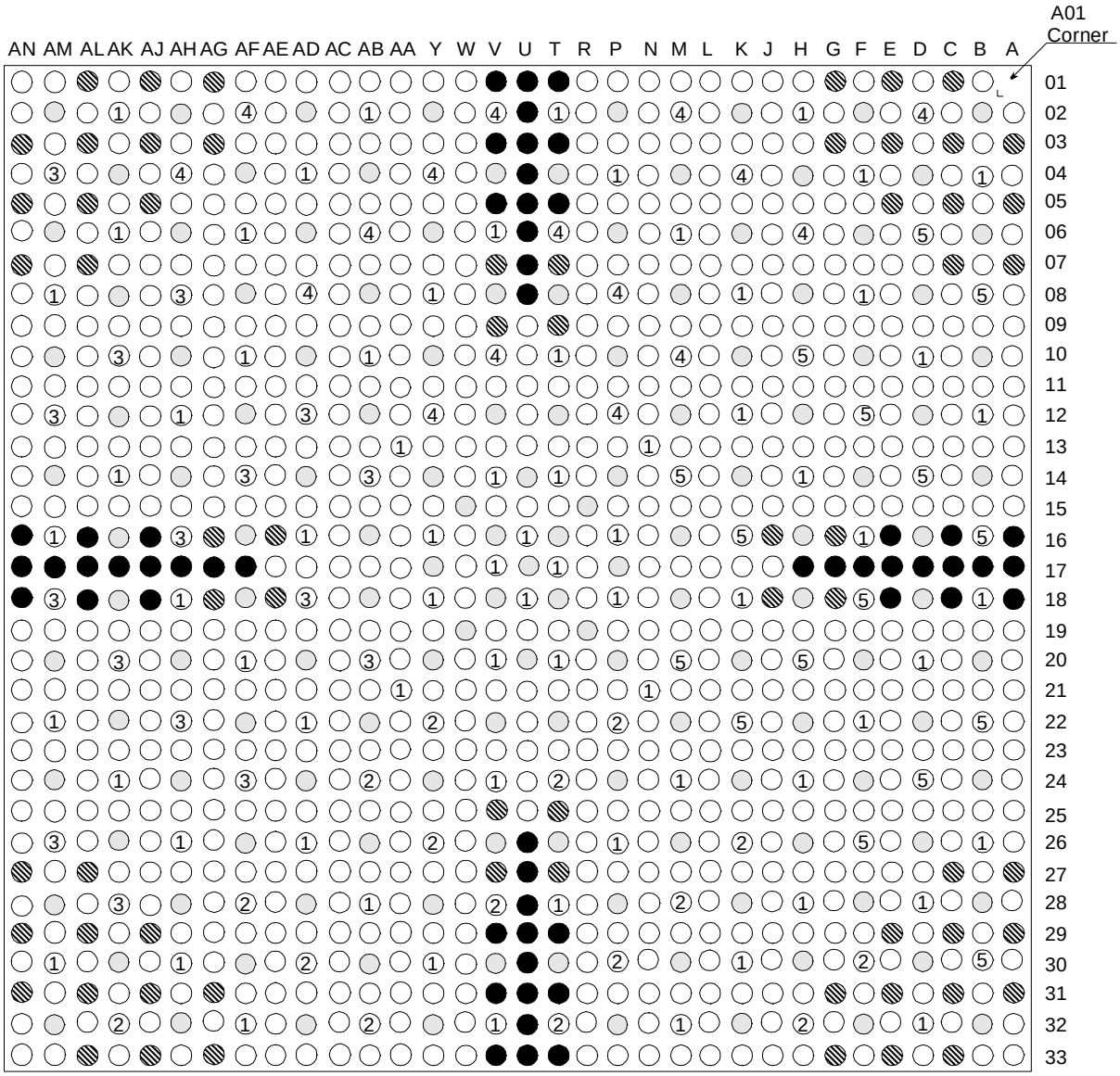


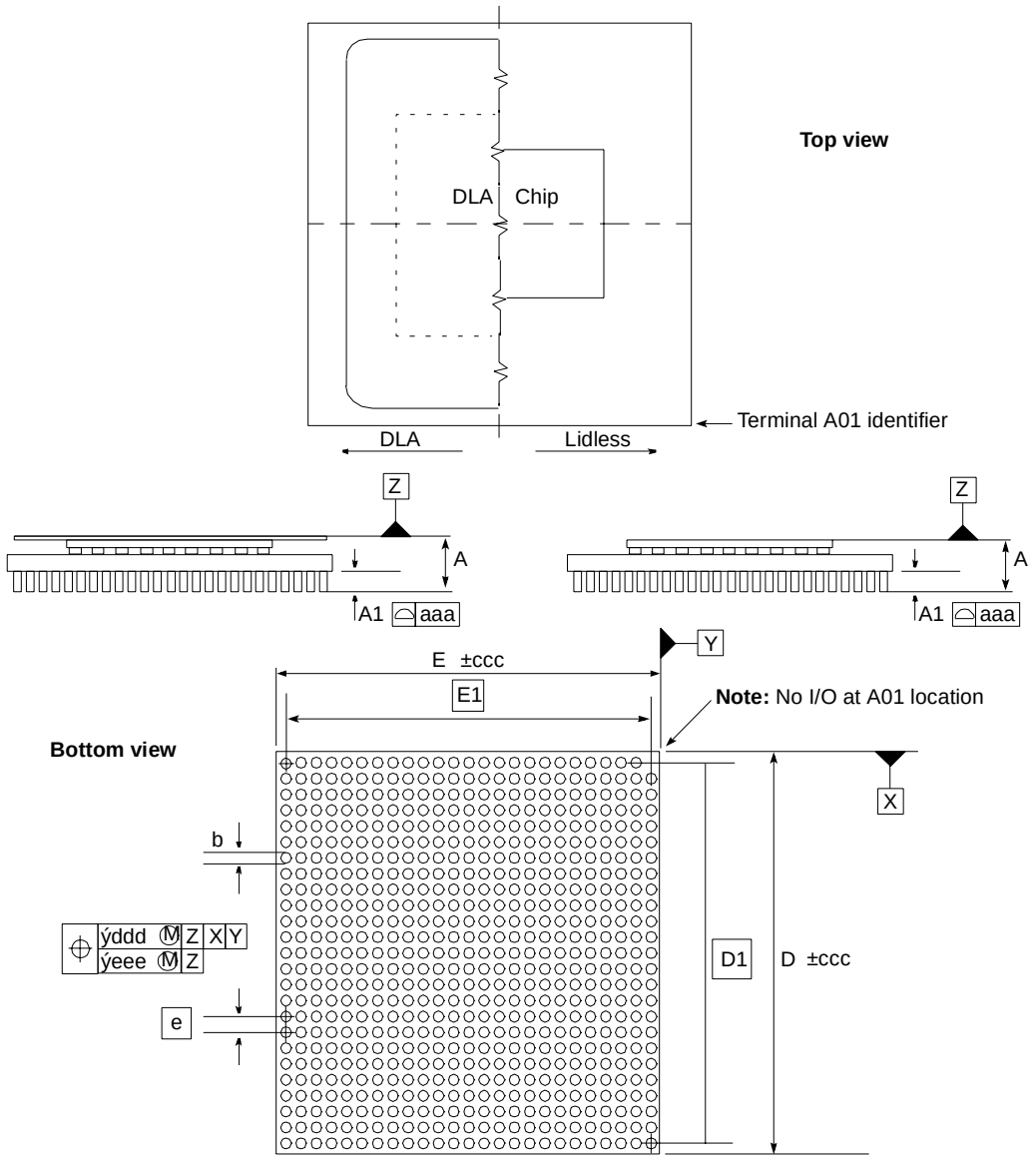
Figure 19: Pins Diagram



Note: For illustrative purposes only. See the IBM ASIC SA-12E Databook available from your IBM representative.

2.2 Mechanical Specifications

Figure 20: Mechanical Diagram



Notes:

1. Refer to *Table 31* on page 70 for mechanical dimensions.
2. Mechanical drawing is not to scale. See your IBM representative for more information.
3. IBM square outline conforms to JEDEC MO-158.

Table 31: Mechanical Specifications

Mechanical Dimensions		Value[1]
A (DLA)	Min[2]	6.23
	Max[3]	6.83
A (Lidless)	Min[2]	4.23
	Max[3]	4.83
A1	Nom	2.21
b	Min	0.48
	Max	0.52
e	Basic	1.27
aaa		0.15
ccc		0.20
ddd		0.30
eee		0.10
D		42.50
D1		40.64
E		42.50
E1		40.64
M[4]		33 x 33
N[5]		1088[5]
Weight (g)		TBD

1. All dimensions are in millimeters, except where noted.
2. Minimum package thickness is calculated using the nominal thickness of all parts. The nominal thickness of an 8-layer package was used for the package thickness.
3. Maximum package thickness is calculated using the nominal thickness of all parts. The nominal thickness of a 12-layer package was used for the package thickness.
4. M = the I/O matrix size.
5. N = the maximum number of I/Os. The number of I/Os shown in the table is the amount after depopulation. Product with 1.27 mm pitch is depopulated by 1 I/O at the A01 corner of the array.
6. IBM square outline conforms to JEDEC MO-158.

2.3 Signal Pin Lists

Note: All unused pins should be left unconnected on the card.

Table 32: Complete Signal Pin Listing by Signal Name (Page 1 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
$\overline{\text{Blade\_Reset}}$	E29	D0_Data(16)	AJ07	D1_Data(01)	AN11
Boot_Picocode	K07	D0_Data(17)	AN04	D1_Data(02)	AL12
Boot_PPC	L04	D0_Data(18)	AN07	D1_Data(03)	AJ10
$\overline{\text{C405\_Debug\_Halt}}$	AB23	D0_Data(19)	AL07	D1_Data(04)	AK09
Clock125	B33	D0_Data(20)	AF09	D1_Data(05)	Y15
Core_Clock	C33	D0_Data(21)	AG08	D1_Data(06)	AA15
D0_Addr(00)	AL10	D0_Data(22)	AE10	D1_Data(07)	AJ11
D0_Addr(01)	AN10	D0_Data(23)	AD11	D1_Data(08)	AK11
D0_Addr(02)	AJ09	D0_Data(24)	AN08	D1_Data(09)	AL13
D0_Addr(03)	AN06	D0_Data(25)	AL09	D1_Data(10)	AN12
D0_Addr(04)	AA14	D0_Data(26)	AL06	D1_Data(11)	AH11
D0_Addr(05)	AB15	D0_Data(27)	AM05	D1_Data(12)	AM13
D0_Addr(06)	AM07	D0_Data(28)	AB13	D1_Data(13)	AN13
D0_Addr(07)	AL08	D0_Data(29)	AC15	D1_Data(14)	AH13
D0_Addr(08)	AM11	D0_Data(30)	AK07	D1_Data(15)	AC16
D0_Addr(09)	AL11	D0_Data(31)	AJ08	D1_DQS(0)	AJ14
D0_Addr(10)	AC14	D0_DQS(0)	AG09	D1_DQS(1)	AN16
D0_Addr(11)	AG10	D0_DQS(1)	AC12	D1_WE	AL16
D0_Addr(12)	AF11	D0_DQS(2)	AE11	D2_Addr(00)	AJ22
D0_CS	AN09	D0_DQS(3)	AH09	D2_Addr(01)	AH21
D0_Data(00)	AA12	D0_WE	AM09	D2_Addr(02)	AN21
D0_Data(01)	AD09	D1_Addr(00)	AB17	D2_Addr(03)	AM21
D0_Data(02)	AG07	D1_Addr(01)	AJ13	D2_Addr(04)	AH23
D0_Data(03)	AB11	D1_Addr(02)	AK13	D2_Addr(05)	AC20
D0_Data(04)	AN02	D1_Addr(03)	AM15	D2_Addr(06)	AD21
D0_Data(05)	AM03	D1_Addr(04)	AN14	D2_Addr(07)	AG23
D0_Data(06)	AN01	D1_Addr(05)	AG12	D2_Addr(08)	AN22
D0_Data(07)	AN03	D1_Addr(06)	AF13	D2_Addr(09)	AL21
D0_Data(08)	AL04	D1_Addr(07)	AE14	D2_Addr(10)	AK23
D0_Data(09)	AG03	D1_Addr(08)	AN15	D2_Addr(11)	AJ23
D0_Data(10)	AH07	D1_Addr(09)	AL14	D2_Addr(12)	AA19
D0_Data(11)	AE09	D1_Addr(10)	W16	D2_CS	AL22
D0_Data(12)	AL03	D1_Addr(11)	AH15	D2_Data(00)	AN18
D0_Data(13)	AC11	D1_Addr(12)	AJ15	D2_Data(01)	AJ19
D0_Data(14)	AJ06	D1_CS	AD15	D2_Data(02)	W18
D0_Data(15)	AK05	D1_Data(00)	AE12	D2_Data(03)	AA18

Table 32: Complete Signal Pin Listing by Signal Name (Page 2 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
D2_Data(04)	AJ20	D3_Data(10)	AG16	D4_Data(16)	E14
D2_Data(05)	AL20	D3_Data(11)	AH17	D4_Data(17)	C14
D2_Data(06)	AN19	D3_Data(12)	AG17	D4_Data(18)	E09
D2_Data(07)	AE20	D3_Data(13)	AG15	D4_Data(19)	A15
D2_Data(08)	AE17	D3_Data(14)	AF15	D4_Data(20)	L15
D2_Data(09)	AE21	D3_Data(15)	AF19	D4_Data(21)	J14
D2_Data(10)	AG22	D3_DQS(0)	AG20	D4_Data(22)	G12
D2_Data(11)	AN20	D3_DQS(1)	AC17	D4_Data(23)	H13
D2_Data(12)	AM19	D3_WE	AD19	D4_Data(24)	A14
D2_Data(13)	AK21	D4_Addr(00)	C12	D4_Data(25)	B15
D2_Data(14)	AJ21	D4_Addr(01)	A11	D4_Data(26)	D13
D2_Data(15)	Y19	D4_Addr(02)	J12	D4_Data(27)	E13
D2_DQS(0)	AB19	D4_Addr(03)	H11	D4_Data(28)	P15
D2_DQS(1)	AK25	D4_Addr(04)	G10	D4_Data(29)	E12
D2_WE	AJ24	D4_Addr(05)	L13	D4_Data(30)	F13
D3_Addr(00)	AG19	D4_Addr(06)	C11	D4_Data(31)	A13
D3_Addr(01)	AJ16	D4_Addr(07)	B11	D4_DQS(0)	D11
D3_Addr(02)	AF17	D4_Addr(08)	D09	D4_DQS(1)	E11
D3_Addr(03)	AJ17	D4_Addr(09)	C08	D4_DQS(2)	N15
D3_Addr(04)	AG18	D4_Addr(10)	M13	D4_DQS(3)	M15
D3_Addr(05)	AE18	D4_Addr(11)	N14	D4_WE	F11
D3_Addr(06)	AA17	D4_Addr(12)	B07	D6_Addr(00)	AL28
D3_Addr(07)	AC18	D4_CS	E10	D6_Addr(01)	AN26
D3_Addr(08)	AK19	D4_Data(00)	M17	D6_Addr(02)	AE24
D3_Addr(09)	AL19	D4_Data(01)	L16	D6_Addr(03)	AG26
D3_Addr(10)	AN17	D4_Data(02)	D15	D6_Addr(04)	AF25
D3_Addr(11)	AK17	D4_Data(03)	C15	D6_Addr(05)	AL27
D3_Addr(12)	AE19	D4_Data(04)	A17	D6_Addr(06)	AN30
D3_CS	AL18	D4_Data(05)	D17	D6_Addr(07)	AJ27
D3_Data(00)	AG13	D4_Data(06)	H09	D6_Addr(08)	AK29
D3_Data(01)	AA16	D4_Data(07)	G14	D6_Addr(09)	AJ28
D3_Data(02)	AG14	D4_Data(08)	G13	D6_Addr(10)	AL29
D3_Data(03)	AE15	D4_Data(09)	K15	D6_Addr(11)	AG21
D3_Data(04)	AL17	D4_Data(10)	C16	D6_Addr(12)	AH27
D3_Data(05)	AM17	D4_Data(11)	A16	D6_ByteEn(0)	AG24
D3_Data(06)	AL15	D4_Data(12)	E15	D6_ByteEn(1)	AF23
D3_Data(07)	AK15	D4_Data(13)	F15	D6_CS	AL31
D3_Data(08)	W17	D4_Data(14)	R17	D6_Data(00)	AL23
D3_Data(09)	AE16	D4_Data(15)	N16	D6_Data(01)	AM23

Table 32: Complete Signal Pin Listing by Signal Name (Page 3 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
D6_Data(02)	AL26	DASL_In_B(0)	E02	$\overline{\text{DASL\_Out\_B(3)}}$	N03
D6_Data(03)	AM27	$\overline{\text{DASL\_In\_B(0)}}$	D01	DASL_Out_B(4)	L05
D6_Data(04)	AB21	DASL_In_B(1)	J02	$\overline{\text{DASL\_Out\_B(4)}}$	L06
D6_Data(05)	AN28	$\overline{\text{DASL\_In\_B(1)}}$	H01	DASL_Out_B(5)	K05
D6_Data(06)	AN24	DASL_In_B(2)	F03	$\overline{\text{DASL\_Out\_B(5)}}$	L07
D6_Data(07)	AL24	$\overline{\text{DASL\_In\_B(2)}}$	F01	DASL_Out_B(6)	J05
D6_Data(08)	AH25	DASL_In_B(3)	G02	$\overline{\text{DASL\_Out\_B(6)}}$	J04
D6_Data(09)	AE23	$\overline{\text{DASL\_In\_B(3)}}$	G04	DASL_Out_B(7)	H05
D6_Data(10)	AC21	DASL_In_B(4)	J03	$\overline{\text{DASL\_Out\_B(7)}}$	H03
D6_Data(11)	AN25	$\overline{\text{DASL\_In\_B(4)}}$	J01	DA_BA(0)	AN29
D6_Data(12)	AJ26	DASL_In_B(5)	K01	DA_BA(1)	AA20
D6_Data(13)	AK27	$\overline{\text{DASL\_In\_B(5)}}$	K03	DA_CAS	AN27
D6_Data(14)	AE22	DASL_In_B(6)	L03	DA_Clk	AM25
D6_Data(15)	AC22	$\overline{\text{DASL\_In\_B(6)}}$	L02	DA_$\overline{\text{Clk}}$	AL25
D6_DQS(0)	AL30	DASL_In_B(7)	N04	DA_RAS	AG25
D6_DQS(1)	AN31	$\overline{\text{DASL\_In\_B(7)}}$	M03	DB_BA(0)	AG11
D6_DQS(2)	AN33	DASL_Out_A(0)	Y01	DB_BA(1)	AD13
D6_DQS(3)	AM31	$\overline{\text{DASL\_Out\_A(0)}}$	W02	DB_CAS	AJ12
D6_DQS_Par(00)	AN32	DASL_Out_A(1)	W03	DB_Clk	AH19
D6_DQS_Par(01)	AF21	$\overline{\text{DASL\_Out\_A(1)}}$	AA01	$\overline{\text{DB\_Clk}}$	AJ18
D6_Parity(00)	AM29	DASL_Out_A(2)	AB01	DB_RAS	AE13
D6_Parity(01)	AN23	$\overline{\text{DASL\_Out\_A(2)}}$	AA02	DC_BA(0)	D25
D6_WE	AJ25	DASL_Out_A(3)	AC06	DC_BA(1)	J17
DASL_In_A(0)	AK01	$\overline{\text{DASL\_Out\_A(3)}}$	AC05	DC_CAS	N19
$\overline{\text{DASL\_In\_A(0)}}$	AJ02	DASL_Out_A(4)	AC07	DC_Clk	E23
DASL_In_A(1)	AF01	$\overline{\text{DASL\_Out\_A(4)}}$	AD05	DC_$\overline{\text{Clk}}$	D23
$\overline{\text{DASL\_In\_A(1)}}$	AE02	DASL_Out_A(5)	AE04	DC_RAS	C21
DASL_In_A(2)	AH01	$\overline{\text{DASL\_Out\_A(5)}}$	AE05	DD_BA(0)	A12
$\overline{\text{DASL\_In\_A(2)}}$	AH03	DASL_Out_A(6)	AF03	DD_BA(1)	J13
DASL_In_A(3)	AE01	$\overline{\text{DASL\_Out\_A(6)}}$	AF05	DD_CAS	G11
$\overline{\text{DASL\_In\_A(3)}}$	AE03	DASL_Out_A(7)	AG04	DD_Clk	C13
DASL_In_A(4)	AD03	$\overline{\text{DASL\_Out\_A(7)}}$	AG02	DD_$\overline{\text{Clk}}$	B13
$\overline{\text{DASL\_In\_A(4)}}$	AD01	DASL_Out_B(0)	R02	DD_RAS	K13
DASL_In_A(5)	AC02	$\overline{\text{DASL\_Out\_B(0)}}$	P01	DE_BA(0)	AM01
$\overline{\text{DASL\_In\_A(5)}}$	AC03	DASL_Out_B(1)	N01	DE_BA(1)	AH05
DASL_In_A(6)	AB03	$\overline{\text{DASL\_Out\_B(1)}}$	R03	DE_CAS	AL02
$\overline{\text{DASL\_In\_A(6)}}$	AA04	DASL_Out_B(2)	N02	DE_Clk	AJ04
DASL_In_A(7)	AA03	$\overline{\text{DASL\_Out\_B(2)}}$	M01	DE_$\overline{\text{Clk}}$	AJ05
$\overline{\text{DASL\_In\_A(7)}}$	Y03	DASL_Out_B(3)	P03	DD_RAS	K13

Table 32: Complete Signal Pin Listing by Signal Name (Page 4 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
DMU_A(00)	V31	DMU_B(08)	R28	DMU_C(16)	M23
DMU_A(01)	V29	DMU_B(09)	R27	DMU_C(17)	N33
DMU_A(02)	V27	DMU_B(10)	R26	DMU_C(18)	N32
DMU_A(03)	W33	DMU_B(11)	R25	DMU_C(19)	N31
DMU_A(04)	W32	DMU_B(12)	R24	DMU_C(20)	N30
DMU_A(05)	W31	DMU_B(13)	R23	DMU_C(21)	N29
DMU_A(06)	W30	DMU_B(14)	T33	DMU_C(22)	N28
DMU_A(07)	W29	DMU_B(15)	T31	DMU_C(23)	N27
DMU_A(08)	W28	DMU_B(16)	T29	DMU_C(24)	N26
DMU_A(09)	W27	DMU_B(17)	T27	DMU_C(25)	N25
DMU_A(10)	W26	DMU_B(18)	R22	DMU_C(26)	N24
DMU_A(11)	W25	DMU_B(19)	T23	DMU_C(27)	N23
DMU_A(12)	W24	DMU_B(20)	V25	DMU_C(28)	P33
DMU_A(13)	W23	DMU_B(21)	U32	DMU_C(29)	P31
DMU_A(14)	Y33	DMU_B(22)	U31	DMU_C(30)	P29
DMU_A(15)	Y31	DMU_B(23)	U30	DMU_D(00)	D33
DMU_A(16)	Y29	DMU_B(24)	U29	DMU_D(01)	D31
DMU_A(17)	Y27	DMU_B(25)	U28	DMU_D(02)	G28
DMU_A(18)	Y25	DMU_B(26)	U27	DMU_D(03)	J29
DMU_A(19)	Y23	DMU_B(27)	U26	DMU_D(04)	E30
DMU_A(20)	AA33	DMU_B(28)	U25	DMU_D(05)	F33
DMU_A(21)	AA32	DMU_B(29)	U24	DMU_D(06)	F31
DMU_A(22)	AA31	DMU_B(30)	V33	DMU_D(07)	F29
DMU_A(23)	AA30	DMU_C(00)	L33	DMU_D(08)	G32
DMU_A(24)	AA29	DMU_C(01)	L32	DMU_D(09)	K25
DMU_A(25)	AA28	DMU_C(02)	L31	DMU_D(10)	G30
DMU_A(26)	AA27	DMU_C(03)	L30	DMU_D(11)	G29
DMU_A(27)	AA26	DMU_C(04)	L29	DMU_D(12)	E32
DMU_A(28)	AA25	DMU_C(05)	L28	DMU_D(13)	H33
DMU_A(29)	AB33	DMU_C(06)	L27	DMU_D(14)	H31
DMU_A(30)	AB31	DMU_C(07)	L26	DMU_D(15)	H29
DMU_B(00)	P27	DMU_C(08)	L25	DMU_D(16)	H27
DMU_B(01)	P25	DMU_C(09)	L24	DMU_D(17)	J33
DMU_B(02)	P23	DMU_C(10)	L23	DMU_D(18)	J32
DMU_B(03)	R33	DMU_C(11)	M33	DMU_D(19)	J31
DMU_B(04)	R32	DMU_C(12)	M31	DMU_D(20)	J30
DMU_B(05)	R31	DMU_C(13)	M29	DMU_D(21)	K27
DMU_B(06)	R30	DMU_C(14)	M27	DMU_D(22)	J28
DMU_B(07)	R29	DMU_C(15)	M25	DMU_D(23)	J27

Table 32: Complete Signal Pin Listing by Signal Name (Page 5 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
DMU_D(24)	J26	DS0_Data(18)	J24	DS1_Data(06)	A21
DMU_D(25)	J25	DS0_Data(19)	K23	DS1_Data(07)	F21
DMU_D(26)	K33	DS0_Data(20)	A26	DS1_Data(08)	E22
DMU_D(27)	K31	DS0_Data(21)	C25	DS1_Data(09)	L18
DMU_D(28)	K29	DS0_Data(22)	C28	DS1_Data(10)	L17
DMU_D(29)	E31	DS0_Data(23)	B29	DS1_Data(11)	E21
DMU_D(30)	G31	DS0_Data(24)	M21	DS1_Data(12)	D21
DS0_Addr(00)	A28	DS0_Data(25)	L19	DS1_Data(13)	B19
DS0_Addr(01)	N20	DS0_Data(26)	D27	DS1_Data(14)	A20
DS0_Addr(02)	M19	DS0_Data(27)	E26	DS1_Data(15)	G22
DS0_Addr(03)	B27	DS0_Data(28)	A25	DS1_Data(16)	J21
DS0_Addr(04)	C26	DS0_Data(29)	B25	DS1_Data(17)	J15
DS0_Addr(05)	B23	DS0_Data(30)	G25	DS1_Data(18)	J20
DS0_Addr(06)	C23	DS0_Data(31)	L22	DS1_Data(19)	A19
DS0_Addr(07)	G24	DS0_DQS(0)	F25	DS1_Data(20)	E18
DS0_Addr(08)	H23	DS0_DQS(1)	C24	DS1_Data(21)	C20
DS0_Addr(09)	J22	DS0_DQS(2)	A24	DS1_Data(22)	E20
DS0_Addr(10)	A23	DS0_DQS(3)	E25	DS1_Data(23)	N18
DS0_Addr(11)	C22	DS0_WE	J23	DS1_Data(24)	F19
DS0_Addr(12)	E24	DS1_Addr(00)	N17	DS1_Data(25)	E19
DS0_CS	A31	DS1_Addr(01)	J18	DS1_Data(26)	A18
DS0_Data(00)	P19	DS1_Addr(02)	G18	DS1_Data(27)	C18
DS0_Data(01)	A32	DS1_Addr(03)	E17	DS1_Data(28)	K19
DS0_Data(02)	B31	DS1_Addr(04)	H17	DS1_Data(29)	G21
DS0_Data(03)	A33	DS1_Addr(05)	G19	DS1_Data(30)	G20
DS0_Data(04)	C30	DS1_Addr(06)	H19	DS1_Data(31)	J19
DS0_Data(05)	C31	DS1_Addr(07)	H15	DS1_DQS(0)	B17
DS0_Data(06)	F27	DS1_Addr(08)	G15	DS1_DQS(1)	C19
DS0_Data(07)	H21	DS1_Addr(09)	G17	DS1_DQS(2)	D19
DS0_Data(08)	C29	DS1_Addr(10)	F17	DS1_DQS(3)	R18
DS0_Data(09)	A29	DS1_Addr(11)	G16	DS1_WE	C17
DS0_Data(10)	E28	DS1_Addr(12)	J16	DUM	A01
DS0_Data(11)	D29	DS1_CS	E16	GND	B10
DS0_Data(12)	E27	DS1_Data(00)	A22	GND	AH32
DS0_Data(13)	A30	DS1_Data(01)	G23	GND	AK04
DS0_Data(14)	A27	DS1_Data(02)	K21	GND	B24
DS0_Data(15)	C27	DS1_Data(03)	L20	GND	AH24
DS0_Data(16)	H25	DS1_Data(04)	F23	GND	AH28
DS0_Data(17)	G26	DS1_Data(05)	B21	GND	AF30

Table 32: Complete Signal Pin Listing by Signal Name (Page 6 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
GND	AF18	GND	AM02	GND	K24
GND	B32	GND	F06	GND	AB04
GND	AF12	GND	AK18	GND	M26
GND	B28	GND	AM10	GND	K06
GND	AF16	GND	F20	GND	AB26
GND	AH20	GND	AM14	GND	K02
GND	B20	GND	AD06	GND	T12
GND	Y17	GND	AK22	GND	M22
GND	W19	GND	AD24	GND	R15
GND	Y20	GND	F10	GND	R19
GND	K20	GND	AD14	GND	D04
GND	F32	GND	T30	GND	H12
GND	W15	GND	F28	GND	AF08
GND	Y02	GND	AH06	GND	D08
GND	V30	GND	M08	GND	V12
GND	AM20	GND	D12	GND	AF04
GND	AH10	GND	H18	GND	T16
GND	AF26	GND	P14	GND	V26
GND	AF22	GND	P24	GND	D22
GND	F02	GND	M30	GND	Y10
GND	AH02	GND	P17	GND	P06
GND	B06	GND	T04	GND	V22
GND	AH14	GND	M12	GND	D26
GND	B02	GND	AB30	GND	Y14
GND	B14	GND	H16	GND	AM24
GND	AD20	GND	K10	GND	V04
GND	D16	GND	H04	GND	AM28
GND	AK26	GND	AB12	GND	V08
GND	U20	GND	H26	GND	V16
GND	AK30	GND	P20	GND	AM32
GND	AD28	GND	AB16	GND	H30
GND	AD10	GND	AB18	GND	K32
GND	Y24	GND	P10	GND	K28
GND	AD32	GND	M04	GND	H08
GND	T18	GND	D18	GND	P02
GND	AD02	GND	AB08	GND	P32
GND	Y32	GND	Y17	GND	D30
GND	AK08	GND	T22	GND	AK16
GND	AK12	GND	AM06	GND	T26

Table 32: Complete Signal Pin Listing by Signal Name (Page 7 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
GND	U14	LU_Clk	AJ03	MC_Grant_A(1)	W21
GND	Y28	LU_Data(00)	U15	MC_Grant_B(0)	R20
GND	F14	LU_Data(01)	U13	MC_Grant_B(1)	P21
GND	V18	LU_Data(02)	T09	MGrant_A(0)	V19
GND	AB22	LU_Data(03)	T07	MGrant_A(1)	U19
GND	Y06	LU_Data(04)	R07	MGrant_B(0)	AG27
GND	K14	LU_Data(05)	R08	MGrant_B(1)	AE25
GND	F24	LU_Data(06)	W06	MG_Clk	J07
GND	H22	LU_Data(07)	W05	MG_Data	J06
GND	T08	LU_Data(08)	U08	MG_nIntr	J08
GND	M18	LU_Data(09)	V07	$\overline{\text{Operational}}$	C32
GND	M16	LU_Data(10)	V09	PCI_AD(00)	AB29
GND	U17	LU_Data(11)	U12	PCI_AD(01)	AB27
GND	P28	LU_Data(12)	V15	PCI_AD(02)	AB25
I_FreeQ_Th	V21	LU_Data(13)	W04	PCI_AD(03)	AC33
JTAG_TCk	AA22	LU_Data(14)	V11	PCI_AD(04)	AC32
JTAG_TDI	W22	LU_Data(15)	W07	PCI_AD(05)	AC31
JTAG_TDO	AA23	LU_Data(16)	W08	PCI_AD(06)	AC30
JTAG_TMS	U22	LU_Data(17)	Y07	PCI_AD(07)	AC29
$\overline{\text{JTAG\_TRst}}$	T25	LU_Data(18)	V13	PCI_AD(08)	AC27
LU_Addr(00)	AA09	LU_Data(19)	W13	PCI_AD(09)	AC26
LU_Addr(01)	Y11	LU_Data(20)	W01	PCI_AD(10)	AC25
LU_Addr(02)	AA10	LU_Data(21)	W09	PCI_AD(11)	AC24
LU_Addr(03)	AB07	LU_Data(22)	Y09	PCI_AD(12)	AD33
LU_Addr(04)	AC09	LU_Data(23)	AA06	PCI_AD(13)	AD31
LU_Addr(05)	AE06	LU_Data(24)	T15	PCI_AD(14)	AD29
LU_Addr(06)	AE07	LU_Data(25)	W10	PCI_AD(15)	AD27
LU_Addr(07)	AC01	LU_Data(26)	W12	PCI_AD(16)	AF29
LU_Addr(08)	R04	LU_Data(27)	W14	PCI_AD(17)	AF27
LU_Addr(09)	AG05	LU_Data(28)	AA07	PCI_AD(18)	AG33
LU_Addr(10)	AG06	LU_Data(29)	AA08	PCI_AD(19)	AG32
LU_Addr(11)	AC04	LU_Data(30)	U01	PCI_AD(20)	AG31
LU_Addr(12)	AD07	LU_Data(31)	W11	PCI_AD(21)	AG30
LU_Addr(13)	AF07	LU_Data(32)	Y05	PCI_AD(22)	AG29
LU_Addr(14)	AB05	LU_Data(33)	R05	PCI_AD(23)	AG28
LU_Addr(15)	AE08	LU_Data(34)	U10	PCI_AD(24)	AH29
LU_Addr(16)	AB09	LU_Data(35)	U03	PCI_AD(25)	AJ33
LU_Addr(17)	AA05	LU_R_$\overline{\text{Wrt}}$	AC08	PCI_AD(26)	AJ32
LU_Addr(18)	AA11	MC_Grant_A(0)	Y21	PCI_AD(27)	AJ31

Table 32: Complete Signal Pin Listing by Signal Name (Page 8 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
PCI_AD(28)	AJ30	SCH_Addr(05)	E04	Spare_Tst_Rcvr(3)	T01
PCI_AD(29)	AJ29	SCH_Addr(06)	H07	Spare_Tst_Rcvr(4)	AL01
PCI_AD(30)	AK33	SCH_Addr(07)	F05	Spare_Tst_Rcvr(5)	G03
PCI_AD(31)	AK31	SCH_Addr(08)	F07	Spare_Tst_Rcvr(6)	V01
PCI_Bus_M_Int	AC23	SCH_Addr(09)	C03	Spare_Tst_Rcvr(7)	V03
PCI_Bus_NM_Int	G27	SCH_Addr(10)	D05	Spare_Tst_Rcvr(8)	T03
PCI_CBE(0)	AC28	SCH_Addr(11)	A02	Spare_Tst_Rcvr(9)	U33
PCI_CBE(1)	AD25	SCH_Addr(12)	C04	Switch_BNA	R21
PCI_CBE(2)	AF31	SCH_Addr(13)	B03	Switch_Clk_A	AN05
PCI_CBE(3)	AH31	SCH_Addr(14)	C02	$\overline{\text{Switch\_Clk\_A}}$	AL05
PCI_Clk	AF33	SCH_Addr(15)	D03	Switch_Clk_B	C05
PCI_DevSel	AE29	SCH_Addr(16)	B01	$\overline{\text{Switch\_Clk\_B}}$	A05
PCI_Frame	AE26	SCH_Addr(17)	C01	Testmode(0)	V05
PCI_Grant	AL32	SCH_Addr(18)	E05	Testmode(1)	U06
PCI_IDSel	AH33	SCH_Clk	C07	Thermal_In	U04
PCI_IntA	AM33	SCH_Data(00)	A10	Thermal_Out	U02
PCI_IRdy	AE27	SCH_Data(01)	C10	Unused	U07
PCI_Par	AE33	SCH_Data(02)	F09	Unused	N05
PCI_PErr	AE31	SCH_Data(03)	J11	Unused	T11
PCI_Request	AL33	SCH_Data(04)	L12	Unused	N10
PCI_SErr	AE32	SCH_Data(05)	G09	Unused	N09
PCI_Speed	M07	SCH_Data(06)	B09	Unused	T13
PCI_Stop	AE30	SCH_Data(07)	A09	Unused	J09
PCI_TRdy	AE28	SCH_Data(08)	A06	Unused	N12
PLLA_GND	AG01	SCH_Data(09)	E08	Unused	R16
PLLA_V_{DD}	AJ01	SCH_Data(10)	G06	Unused	N07
PLLB_GND	G01	SCH_Data(11)	G07	Unused	M11
PLLB_V_{DD}	E01	SCH_Data(12)	C06	Unused	R12
PLLC_GND	G33	SCH_Data(13)	D07	Unused	R11
PLLC_V_{DD}	E33	SCH_Data(14)	C09	Unused	N06
RES_Data	T21	SCH_Data(15)	A08	Unused	R09
RES_Sync	U21	SCH_Data(16)	J10	Unused	U11
Rx_LByte(0)	U23	SCH_Data(17)	G08	Unused	R13
Rx_LByte(1)	V23	SCH_R_$\overline{\text{Wrt}}$	G05	Unused	U09
SCH_Addr(00)	A07	Send_Grant_A	W20	Unused	T05
SCH_Addr(01)	B05	Send_Grant_B	T19	Unused	K09
SCH_Addr(02)	A04	Spare_Tst_Rcvr(0)	U05	Unused	P07
SCH_Addr(03)	E06	Spare_Tst_Rcvr(1)	E03	Unused	L08
SCH_Addr(04)	E07	Spare_Tst_Rcvr(2)	A03	Unused	L09

Table 32: Complete Signal Pin Listing by Signal Name (Page 9 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
Unused	R01	V_{DD}	F04	V_{DD}	V14
Unused	P13	V_{DD}	B12	V_{DD}	V06
Unused	M09	V_{DD}	AB10	V_{DD}	AD16
Unused	N08	V_{DD}	AH26	V_{DD}	D28
Unused	L01	V_{DD}	AK06	V_{DD}	B18
Unused	N22	V_{DD}	AK02	V_{DD}	B26
Unused	M05	V_{DD}	M32	V_{DD}	AF10
Unused	AA24	V_{DD}	Y18	V_{DD}	AM16
Unused	R14	V_{DD}	K30	V_{DD}	U18
Unused	L11	V_{DD}	V32	VRef1(0)	AD23
Unused	R10	V_{DD}	V24	VRef1(1)	K11
Unused	P05	V_{DD}	T10	VRef1(2)	AC10
Unused	R06	V_{DD}	AM22	VRef2(0)	AC19
Unused	P11	V_{DD}	AF20	VRef2(1)	AD17
Unused	P09	V_{DD}	D32	VRef2(2)	AC13
V_{DD}	AD04	V_{DD}	P16	VRef2(3)	L14
V_{DD}	T20	V_{DD}	Y16	VRef2(4)	K17
V_{DD}	AF06	V_{DD}	AD26	VRef2(5)	L21
V_{DD}	AB28	V_{DD}	B04	VRef2(6)	Y13
V_{DD}	AA13	V_{DD}	AM30	VRef2(7)	N11
V_{DD}	K12	V_{DD}	Y30	VRef2(8)	L10
V_{DD}	T17	V_{DD}	AH30	2.5 V	P12
V_{DD}	H02	V_{DD}	Y08	2.5 V	D02
V_{DD}	N21	V_{DD}	F08	2.5 V	AK10
V_{DD}	K08	V_{DD}	P04	2.5 V	F26
V_{DD}	T14	V_{DD}	AK24	2.5 V	H10
V_{DD}	T02	V_{DD}	P18	2.5 V	D06
V_{DD}	T28	V_{DD}	AH18	2.5 V	AM18
V_{DD}	K18	V_{DD}	M24	2.5 V	D14
V_{DD}	H28	V_{DD}	AF32	2.5 V	AM12
V_{DD}	F22	V_{DD}	AM08	2.5 V	AF02
V_{DD}	V17	V_{DD}	AK14	2.5 V	F18
V_{DD}	V20	V_{DD}	AH12	2.5 V	K04
V_{DD}	H24	V_{DD}	M06	2.5 V	AB20
V_{DD}	P26	V_{DD}	H14	2.5 V	Y04
V_{DD}	U16	V_{DD}	F16	2.5 V	AH22
V_{DD}	D10	V_{DD}	N13	2.5 V	AH04
V_{DD}	D20	V_{DD}	AB02	2.5 V	V10
V_{DD}	AA21	V_{DD}	AD22	2.5 V	Y12

Table 32: Complete Signal Pin Listing by Signal Name (Page 10 of 10)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
2.5 V	M10	2.5 V	M02	3.3 V	Y22
2.5 V	AH08	2.5 V	B22	3.3 V	M28
2.5 V	AD18	2.5 V	AH16	3.3 V	Y26
2.5 V	V02	2.5 V	AM26	3.3 V	P30
2.5 V	D24	2.5 V	AF24	3.3 V	T24
2.5 V	T06	2.5 V	AB06	3.3 V	P22
2.5 V	B16	2.5 V	AM04	3.3 V	AF28
2.5 V	H06	2.5 V	F12	3.3 V	AB24
2.5 V	AF14	2.5 V	B08	3.3 V	F30
2.5 V	M14	2.5 V	AK20	3.3 V	AB32
2.5 V	AB14	2.5 V	K16	3.3 V	V28
2.5 V	B30	2.5 V	AK28	3.3 V	H32
2.5 V	H20	2.5 V	P08	3.3 V	K26
2.5 V	K22	2.5 V	M20	3.3 V	AK32
2.5 V	AD08	2.5 V	AD12	3.3 V	T32
				3.3 V	AD30

Note: All unused pins should be left unused on the card.

Table 33: Complete Signal Pin Listing by Grid Position (Page 1 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
A01	DUM	AA04	$\overline{\text{DASL\_In\_A(6)}}$	AB07	LU_Addr(03)
A02	SCH_Addr(11)	AA05	LU_Addr(17)	AB08	GND
A03	Spare_Tst_Rcvr(2)	AA06	LU_Data(23)	AB09	LU_Addr(16)
A04	SCH_Addr(02)	AA07	LU_Data(28)	AB10	V_{DD}
A05	$\overline{\text{Switch\_Clk\_B}}$	AA08	LU_Data(29)	AB11	D0_Data(03)
A06	SCH_Data(08)	AA09	LU_Addr(00)	AB12	GND
A07	SCH_Addr(00)	AA10	LU_Addr(02)	AB13	D0_Data(28)
A08	SCH_Data(15)	AA11	LU_Addr(18)	AB14	2.5V
A09	SCH_Data(07)	AA12	D0_Data(00)	AB15	D0_Addr(05)
A10	SCH_Data(00)	AA13	V_{DD}	AB16	GND
A11	D4_Addr(01)	AA14	D0_Addr(04)	AB17	D1_Addr(00)
A12	DD_BA(0)	AA15	D1_Data(06)	AB18	GND
A13	D4_Data(31)	AA16	D3_Data(01)	AB19	D2_DQS(0)
A14	D4_Data(24)	AA17	D3_Addr(06)	AB20	2.5V
A15	D4_Data(19)	AA18	D2_Data(03)	AB21	D6_Data(04)
A16	D4_Data(11)	AA19	D2_Addr(12)	AB22	GND
A17	D4_Data(04)	AA20	DA_BA(1)	AB23	$\overline{\text{C405\_Debug\_Halt}}$
A18	DS1_Data(26)	AA21	V_{DD}	AB24	3.3V
A19	DS1_Data(19)	AA22	JTAG_TCk	AB25	PCI_AD(02)
A20	DS1_Data(14)	AA23	JTAG_TDO	AB26	GND
A21	DS1_Data(06)	AA24	Unused	AB27	PCI_AD(01)
A22	DS1_Data(00)	AA25	DMU_A(28)	AB28	V_{DD}
A23	DS0_Addr(10)	AA26	DMU_A(27)	AB29	PCI_AD(00)
A24	DS0_DQS(2)	AA27	DMU_A(26)	AB30	GND
A25	DS0_Data(28)	AA28	DMU_A(25)	AB31	DMU_A(30)
A26	DS0_Data(20)	AA29	DMU_A(24)	AB32	3.3V
A27	DS0_Data(14)	AA30	DMU_A(23)	AB33	DMU_A(29)
A28	DS0_Addr(00)	AA31	DMU_A(22)	AC01	LU_Addr(07)
A29	DS0_Data(09)	AA32	DMU_A(21)	AC02	DASL_In_A(5)
A30	DS0_Data(13)	AA33	DMU_A(20)	AC03	$\overline{\text{DASL\_In\_A(5)}}$
A31	DS0_CS	AB01	DASL_Out_A(2)	AC04	LU_Addr(11)
A32	DS0_Data(01)	AB02	V_{DD}	AC05	$\overline{\text{DASL\_Out\_A(3)}}$
A33	DS0_Data(03)	AB03	DASL_In_A(6)	AC06	DASL_Out_A(3)
AA01	$\overline{\text{DASL\_Out\_A(1)}}$	AB04	GND	AC07	DASL_Out_A(4)
AA02	$\overline{\text{DASL\_Out\_A(2)}}$	AB05	LU_Addr(14)	AC08	LU_R_$\overline{\text{Wrt}}$
AA03	DASL_In_A(7)	AB06	2.5V	AC09	LU_Addr(04)

Table 33: Complete Signal Pin Listing by Grid Position (Page 2 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
AC10	VRef1(2)	AD13	DB_BA(1)	AE16	D3_Data(09)
AC11	D0_Data(13)	AD14	GND	AE17	D2_Data(08)
AC12	D0_DQS(1)	AD15	D1_CS	AE18	D3_Addr(05)
AC13	VRef2(2)	AD16	V_{DD}	AE19	D3_Addr(12)
AC14	D0_Addr(10)	AD17	VRef2(1)	AE20	D2_Data(07)
AC15	D0_Data(29)	AD18	2.5V	AE21	D2_Data(09)
AC16	D1_Data(15)	AD19	D3_WE	AE22	D6_Data(14)
AC17	D3_DQS(1)	AD20	GND	AE23	D6_Data(09)
AC18	D3_Addr(07)	AD21	D2_Addr(06)	AE24	D6_Addr(02)
AC19	VRef2(0)	AD22	V_{DD}	AE25	MGrant_B(1)
AC20	D2_Addr(05)	AD23	VRef1(0)	AE26	PCI_Frame
AC21	D6_Data(10)	AD24	GND	AE27	PCI_IRdy
AC22	D6_Data(15)	AD25	PCI_CBE(1)	AE28	PCI_TRdy
AC23	PCI_Bus_M_Int	AD26	V_{DD}	AE29	PCI_DevSel
AC24	PCI_AD(11)	AD27	PCI_AD(15)	AE30	PCI_Stop
AC25	PCI_AD(10)	AD28	GND	AE31	PCI_PErr
AC26	PCI_AD(09)	AD29	PCI_AD(14)	AE32	PCI_SErr
AC27	PCI_AD(08)	AD30	3.3V	AE33	PCI_Par
AC28	PCI_CBE(0)	AD31	PCI_AD(13)	AF01	DASL_In_A(1)
AC29	PCI_AD(07)	AD32	GND	AF02	2.5V
AC30	PCI_AD(06)	AD33	PCI_AD(12)	AF03	DASL_Out_A(6)
AC31	PCI_AD(05)	AE01	DASL_In_A(3)	AF04	GND
AC32	PCI_AD(04)	AE02	$\overline{\text{DASL\_In\_A(1)}}$	AF05	$\overline{\text{DASL\_Out\_A(6)}}$
AC33	PCI_AD(03)	AE03	$\overline{\text{DASL\_In\_A(3)}}$	AF06	V_{DD}
AD01	$\overline{\text{DASL\_In\_A(4)}}$	AE04	DASL_Out_A(5)	AF07	LU_Addr(13)
AD02	GND	AE05	$\overline{\text{DASL\_Out\_A(5)}}$	AF08	GND
AD03	DASL_In_A(4)	AE06	LU_Addr(05)	AF09	D0_Data(20)
AD04	V_{DD}	AE07	LU_Addr(06)	AF10	V_{DD}
AD05	$\overline{\text{DASL\_Out\_A(4)}}$	AE08	LU_Addr(15)	AF11	D0_Addr(12)
AD06	GND	AE09	D0_Data(11)	AF12	GND
AD07	LU_Addr(12)	AE10	D0_Data(22)	AF13	D1_Addr(06)
AD08	2.5V	AE11	D0_DQS(2)	AF14	2.5V
AD09	D0_Data(01)	AE12	D1_Data(00)	AF15	D3_Data(14)
AD10	GND	AE13	DB_RAS	AF16	GND
AD11	D0_Data(23)	AE14	D1_Addr(07)	AF17	D3_Addr(02)
AD12	2.5V	AE15	D3_Data(03)	AF18	GND
AF19	D3_Data(15)	AG22	D2_Data(10)	AH25	D6_Data(08)

Table 33: Complete Signal Pin Listing by Grid Position (Page 3 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
AF20	V_{DD}	AG23	D2_Addr(07)	AH26	V_{DD}
AF21	D6_DQS_Par(01)	AG24	D6_ByteEn(0)	AH27	D6_Addr(12)
AF22	GND	AG25	DA_RAS	AH28	GND
AF23	D6_ByteEn(1)	AG26	D6_Addr(03)	AH29	PCI_AD(24)
AF24	2.5V	AG27	MGrant_B(0)	AH30	V_{DD}
AF25	D6_Addr(04)	AG28	PCI_AD(23)	AH31	PCI_CBE(3)
AF26	GND	AG29	PCI_AD(22)	AH32	GND
AF27	PCI_AD(17)	AG30	PCI_AD(21)	AH33	PCI_IDSel
AF28	3.3V	AG31	PCI_AD(20)	AJ01	PLLA_V_{DD}
AF29	PCI_AD(16)	AG32	PCI_AD(19)	AJ02	$\overline{\text{DASL\_In\_A(0)}}$
AF30	GND	AG33	PCI_AD(18)	AJ03	LU_Clk
AF31	PCI_CBE(2)	AH01	DASL_In_A(2)	AJ04	DE_Clk
AF32	V_{DD}	AH02	GND	AJ05	DE_$\overline{\text{Clk}}$
AF33	PCI_Clk	AH03	$\overline{\text{DASL\_In\_A(2)}}$	AJ06	D0_Data(14)
AG01	PLLA_GND	AH04	2.5V	AJ07	D0_Data(16)
AG02	$\overline{\text{DASL\_Out\_A(7)}}$	AH05	DE_BA(1)	AJ08	D0_Data(31)
AG03	D0_Data(09)	AH06	GND	AJ09	D0_Addr(02)
AG04	DASL_Out_A(7)	AH07	D0_Data(10)	AJ10	D1_Data(03)
AG05	LU_Addr(09)	AH08	2.5V	AJ11	D1_Data(07)
AG06	LU_Addr(10)	AH09	D0_DQS(3)	AJ12	DB_CAS
AG07	D0_Data(02)	AH10	GND	AJ13	D1_Addr(01)
AG08	D0_Data(21)	AH11	D1_Data(11)	AJ14	D1_DQS(0)
AG09	D0_DQS(0)	AH12	V_{DD}	AJ15	D1_Addr(12)
AG10	D0_Addr(11)	AH13	D1_Data(14)	AJ16	D3_Addr(01)
AG11	DB_BA(0)	AH14	GND	AJ17	D3_Addr(03)
AG12	D1_Addr(05)	AH15	D1_Addr(11)	AJ18	DB_$\overline{\text{Clk}}$
AG13	D3_Data(00)	AH16	2.5V	AJ19	D2_Data(01)
AG14	D3_Data(02)	AH17	D3_Data(11)	AJ20	D2_Data(04)
AG15	D3_Data(13)	AH18	V_{DD}	AJ21	D2_Data(14)
AG16	D3_Data(10)	AH19	DB_Clk	AJ22	D2_Addr(00)
AG17	D3_Data(12)	AH20	GND	AJ23	D2_Addr(11)
AG18	D3_Addr(04)	AH21	D2_Addr(01)	AJ24	D2_WE
AG19	D3_Addr(00)	AH22	2.5V	AJ25	D6_WE
AG20	D3_DQS(0)	AH23	D2_Addr(04)	AJ26	D6_Data(12)
AG21	D6_Addr(11)	AH24	GND	AJ27	D6_Addr(07)
AJ28	D6_Addr(09)	AK31	PCI_AD(31)	AM01	DE_BA(0)
AJ29	PCI_AD(29)	AK32	3.3V	AM02	GND

Table 33: Complete Signal Pin Listing by Grid Position (Page 4 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
AJ30	PCI_AD(28)	AK33	PCI_AD(30)	AM03	D0_Data(05)
AJ31	PCI_AD(27)	AL01	Spare_Tst_Rcvr(4)	AM04	2.5V
AJ32	PCI_AD(26)	AL02	DE_CAS	AM05	D0_Data(27)
AJ33	PCI_AD(25)	AL03	D0_Data(12)	AM06	GND
AK01	DASL_In_A(0)	AL04	D0_Data(08)	AM07	D0_Addr(06)
AK02	V_{DD}	AL05	$\overline{\text{Switch\_Clk\_A}}$	AM08	V_{DD}
AK03	DE_RAS	AL06	D0_Data(26)	AM09	D0_WE
AK04	GND	AL07	D0_Data(19)	AM10	GND
AK05	D0_Data(15)	AL08	D0_Addr(07)	AM11	D0_Addr(08)
AK06	V_{DD}	AL09	D0_Data(25)	AM12	2.5V
AK07	D0_Data(30)	AL10	D0_Addr(00)	AM13	D1_Data(12)
AK08	GND	AL11	D0_Addr(09)	AM14	GND
AK09	D1_Data(04)	AL12	D1_Data(02)	AM15	D1_Addr(03)
AK10	2.5V	AL13	D1_Data(09)	AM16	V_{DD}
AK11	D1_Data(08)	AL14	D1_Addr(09)	AM17	D3_Data(05)
AK12	GND	AL15	D3_Data(06)	AM18	2.5V
AK13	D1_Addr(02)	AL16	D1_WE	AM19	D2_Data(12)
AK14	V_{DD}	AL17	D3_Data(04)	AM20	GND
AK15	D3_Data(07)	AL18	D3_CS	AM21	D2_Addr(03)
AK16	GND	AL19	D3_Addr(09)	AM22	V_{DD}
AK17	D3_Addr(11)	AL20	D2_Data(05)	AM23	D6_Data(01)
AK18	GND	AL21	D2_Addr(09)	AM24	GND
AK19	D3_Addr(08)	AL22	D2_CS	AM25	DA_Clk
AK20	2.5V	AL23	D6_Data(00)	AM26	2.5V
AK21	D2_Data(13)	AL24	D6_Data(07)	AM27	D6_Data(03)
AK22	GND	AL25	DA_$\overline{\text{Clk}}$	AM28	GND
AK23	D2_Addr(10)	AL26	D6_Data(02)	AM29	D6_Parity(00)
AK24	V_{DD}	AL27	D6_Addr(05)	AM30	V_{DD}
AK25	D2_DQS(1)	AL28	D6_Addr(00)	AM31	D6_DQS(3)
AK26	GND	AL29	D6_Addr(10)	AM32	GND
AK27	D6_Data(13)	AL30	D6_DQS(0)	AM33	PCI_IntA
AK28	2.5V	AL31	D6_CS	AN01	D0_Data(06)
AK29	D6_Addr(08)	AL32	PCI_Grant	AN02	D0_Data(04)
AK30	GND	AL33	PCI_Request	AN03	D0_Data(07)
AN04	D0_Data(17)	B07	D4_Addr(12)	C10	SCH_Data(01)
AN05	Switch_Clk_A	B08	2.5V	C11	D4_Addr(06)
AN06	D0_Addr(03)	B09	SCH_Data(06)	C12	D4_Addr(00)

Table 33: Complete Signal Pin Listing by Grid Position (Page 5 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
AN07	D0_Data(18)	B10	GND	C13	DD_Clk
AN08	D0_Data(24)	B11	D4_Addr(07)	C14	D4_Data(17)
AN09	D0_CS	B12	V_{DD}	C15	D4_Data(03)
AN10	D0_Addr(01)	B13	DD_$\overline{Clk}$	C16	D4_Data(10)
AN11	D1_Data(01)	B14	GND	C17	DS1_WE
AN12	D1_Data(10)	B15	D4_Data(25)	C18	DS1_Data(27)
AN13	D1_Data(13)	B16	2.5V	C19	DS1_DQS(1)
AN14	D1_Addr(04)	B17	DS1_DQS(0)	C20	DS1_Data(21)
AN15	D1_Addr(08)	B18	V_{DD}	C21	DC_RAS
AN16	D1_DQS(1)	B19	DS1_Data(13)	C22	DS0_Addr(11)
AN17	D3_Addr(10)	B20	GND	C23	DS0_Addr(06)
AN18	D2_Data(00)	B21	DS1_Data(05)	C24	DS0_DQS(1)
AN19	D2_Data(06)	B22	2.5V	C25	DS0_Data(21)
AN20	D2_Data(11)	B23	DS0_Addr(05)	C26	DS0_Addr(04)
AN21	D2_Addr(02)	B24	GND	C27	DS0_Data(15)
AN22	D2_Addr(08)	B25	DS0_Data(29)	C28	DS0_Data(22)
AN23	D6_Parity(01)	B26	V_{DD}	C29	DS0_Data(08)
AN24	D6_Data(06)	B27	DS0_Addr(03)	C30	DS0_Data(04)
AN25	D6_Data(11)	B28	GND	C31	DS0_Data(05)
AN26	D6_Addr(01)	B29	DS0_Data(23)	C32	$\overline{Operational}$
AN27	DA_CAS	B30	2.5V	C33	Core_Clock
AN28	D6_Data(05)	B31	DS0_Data(02)	D01	$\overline{DASL\_In\_B(0)}$
AN29	DA_BA(0)	B32	GND	D02	2.5V
AN30	D6_Addr(06)	B33	Clock125	D03	SCH_Addr(15)
AN31	D6_DQS(1)	C01	SCH_Addr(17)	D04	GND
AN32	D6_DQS_Par(00)	C02	SCH_Addr(14)	D05	SCH_Addr(10)
AN33	D6_DQS(2)	C03	SCH_Addr(09)	D06	2.5V
B01	SCH_Addr(16)	C04	SCH_Addr(12)	D07	SCH_Data(13)
B02	GND	C05	Switch_Clk_B	D08	GND
B03	SCH_Addr(13)	C06	SCH_Data(12)	D09	D4_Addr(08)
B04	V_{DD}	C07	SCH_Clk	D10	V_{DD}
B05	SCH_Addr(01)	C08	D4_Addr(09)	D11	D4_DQS(0)
B06	GND	C09	SCH_Data(14)	D12	GND
D13	D4_Data(26)	E16	DS1_CS	F19	DS1_Data(24)
D14	2.5V	E17	DS1_Addr(03)	F20	GND
D15	D4_Data(02)	E18	DS1_Data(20)	F21	DS1_Data(07)
D16	GND	E19	DS1_Data(25)	F22	V_{DD}

Table 33: Complete Signal Pin Listing by Grid Position (Page 6 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
D17	D4_Data(05)	E20	DS1_Data(22)	F23	DS1_Data(04)
D18	GND	E21	DS1_Data(11)	F24	GND
D19	DS1_DQS(2)	E22	DS1_Data(08)	F25	DS0_DQS(0)
D20	V_{DD}	E23	DC_Clk	F26	2.5V
D21	DS1_Data(12)	E24	DS0_Addr(12)	F27	DS0_Data(06)
D22	GND	E25	DS0_DQS(3)	F28	GND
D23	DC_$\overline{\text{Clk}}$	E26	DS0_Data(27)	F29	DMU_D(07)
D24	2.5V	E27	DS0_Data(12)	F30	3.3V
D25	DC_BA(0)	E28	DS0_Data(10)	F31	DMU_D(06)
D26	GND	E29	$\overline{\text{Blade\_Reset}}$	F32	GND
D27	DS0_Data(26)	E30	DMU_D(04)	F33	DMU_D(05)
D28	V_{DD}	E31	DMU_D(29)	G01	PLLB_GND
D29	DS0_Data(11)	E32	DMU_D(12)	G02	DASL_In_B(3)
D30	GND	E33	PLLC_V_{DD}	G03	Spare_Tst_Rcvr(5)
D31	DMU_D(01)	F01	$\overline{\text{DASL\_In\_B(2)}}$	G04	$\overline{\text{DASL\_In\_B(3)}}$
D32	V_{DD}	F02	GND	G05	SCH_R_$\overline{\text{Wrt}}$
D33	DMU_D(00)	F03	DASL_In_B(2)	G06	SCH_Data(10)
E01	PLLB_V_{DD}	F04	V_{DD}	G07	SCH_Data(11)
E02	DASL_In_B(0)	F05	SCH_Addr(07)	G08	SCH_Data(17)
E03	Spare_Tst_Rcvr(1)	F06	GND	G09	SCH_Data(05)
E04	SCH_Addr(05)	F07	SCH_Addr(08)	G10	D4_Addr(04)
E05	SCH_Addr(18)	F08	V_{DD}	G11	DD_CAS
E06	SCH_Addr(03)	F09	SCH_Data(02)	G12	D4_Data(22)
E07	SCH_Addr(04)	F10	GND	G13	D4_Data(08)
E08	SCH_Data(09)	F11	D4_WE	G14	D4_Data(07)
E09	D4_Data(18)	F12	2.5V	G15	DS1_Addr(08)
E10	D4_CS	F13	D4_Data(30)	G16	DS1_Addr(11)
E11	D4_DQS(1)	F14	GND	G17	DS1_Addr(09)
E12	D4_Data(29)	F15	D4_Data(13)	G18	DS1_Addr(02)
E13	D4_Data(27)	F16	V_{DD}	G19	DS1_Addr(05)
E14	D4_Data(16)	F17	DS1_Addr(10)	G20	DS1_Data(30)
E15	D4_Data(12)	F18	2.5V	G21	DS1_Data(29)
G22	DS1_Data(15)	H25	DS0_Data(16)	J28	DMU_D(22)
G23	DS1_Data(01)	H26	GND	J29	DMU_D(03)
G24	DS0_Addr(07)	H27	DMU_D(16)	J30	DMU_D(20)
G25	DS0_Data(30)	H28	V_{DD}	J31	DMU_D(19)
G26	DS0_Data(17)	H29	DMU_D(15)	J32	DMU_D(18)

Table 33: Complete Signal Pin Listing by Grid Position (Page 7 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
G27	PCI_Bus_NM_Int	H30	GND	J33	DMU_D(17)
G28	DMU_D(02)	H31	DMU_D(14)	K01	DASL_In_B(5)
G29	DMU_D(11)	H32	3.3V	K02	GND
G30	DMU_D(10)	H33	DMU_D(13)	K03	$\overline{\text{DASL\_In\_B(5)}}$
G31	DMU_D(30)	J01	$\overline{\text{DASL\_In\_B(4)}}$	K04	2.5V
G32	DMU_D(08)	J02	DASL_In_B(1)	K05	DASL_Out_B(5)
G33	PLLC_GND	J03	DASL_In_B(4)	K06	GND
H01	$\overline{\text{DASL\_In\_B(1)}}$	J04	$\overline{\text{DASL\_Out\_B(6)}}$	K07	Boot_Picocode
H02	V_{DD}	J05	DASL_Out_B(6)	K08	V_{DD}
H03	$\overline{\text{DASL\_Out\_B(7)}}$	J06	MG_Data	K09	Unused
H04	GND	J07	MG_Clk	K10	GND
H05	DASL_Out_B(7)	J08	MG_nIntr	K11	VRef1(1)
H06	2.5V	J09	Unused	K12	V_{DD}
H07	SCH_Addr(06)	J10	SCH_Data(16)	K13	DD_RAS
H08	GND	J11	SCH_Data(03)	K14	GND
H09	D4_Data(06)	J12	D4_Addr(02)	K15	D4_Data(09)
H10	2.5V	J13	DD_BA(1)	K16	2.5V
H11	D4_Addr(03)	J14	D4_Data(21)	K17	VRef2(4)
H12	GND	J15	DS1_Data(17)	K18	V_{DD}
H13	D4_Data(23)	J16	DS1_Addr(12)	K19	DS1_Data(28)
H14	V_{DD}	J17	DC_BA(1)	K20	GND
H15	DS1_Addr(07)	J18	DS1_Addr(01)	K21	DS1_Data(02)
H16	GND	J19	DS1_Data(31)	K22	2.5V
H17	DS1_Addr(04)	J20	DS1_Data(18)	K23	DS0_Data(19)
H18	GND	J21	DS1_Data(16)	K24	GND
H19	DS1_Addr(06)	J22	DS0_Addr(09)	K25	DMU_D(09)
H20	2.5V	J23	DS0_WE	K26	3.3V
H21	DS0_Data(07)	J24	DS0_Data(18)	K27	DMU_D(21)
H22	GND	J25	DMU_D(25)	K28	GND
H23	DS0_Addr(08)	J26	DMU_D(24)	K29	DMU_D(28)
H24	V_{DD}	J27	DMU_D(23)	K30	V_{DD}
K31	DMU_D(27)	M01	$\overline{\text{DASL\_Out\_B(2)}}$	N04	DASL_In_B(7)
K32	GND	M02	2.5V	N05	Unused
K33	DMU_D(26)	M03	$\overline{\text{DASL\_In\_B(7)}}$	N06	Unused
L01	Unused	M04	GND	N07	Unused
L02	$\overline{\text{DASL\_In\_B(6)}}$	M05	Unused	N08	Unused
L03	DASL_In_B(6)	M06	V_{DD}	N09	Unused

Table 33: Complete Signal Pin Listing by Grid Position (Page 8 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
L04	Boot_PPC	M07	PCI_Speed	N10	Unused
L05	DASL_Out_B(4)	M08	GND	N11	VRef2(7)
L06	$\overline{\text{DASL\_Out\_B(4)}}$	M09	Unused	N12	Unused
L07	$\overline{\text{DASL\_Out\_B(5)}}$	M10	2.5V	N13	V_{DD}
L08	Unused	M11	Unused	N14	D4_Addr(11)
L09	Unused	M12	GND	N15	D4_DQS(2)
L10	VRef2(8)	M13	D4_Addr(10)	N16	D4_Data(15)
L11	Unused	M14	2.5V	N17	DS1_Addr(00)
L12	SCH_Data(04)	M15	D4_DQS(3)	N18	DS1_Data(23)
L13	D4_Addr(05)	M16	GND	N19	DC_CAS
L14	VRef2(3)	M17	D4_Data(00)	N20	DS0_Addr(01)
L15	D4_Data(20)	M18	GND	N21	V_{DD}
L16	D4_Data(01)	M19	DS0_Addr(02)	N22	Unused
L17	DS1_Data(10)	M20	2.5V	N23	DMU_C(27)
L18	DS1_Data(09)	M21	DS0_Data(24)	N24	DMU_C(26)
L19	DS0_Data(25)	M22	GND	N25	DMU_C(25)
L20	DS1_Data(03)	M23	DMU_C(16)	N26	DMU_C(24)
L21	VRef2(5)	M24	V_{DD}	N27	DMU_C(23)
L22	DS0_Data(31)	M25	DMU_C(15)	N28	DMU_C(22)
L23	DMU_C(10)	M26	GND	N29	DMU_C(21)
L24	DMU_C(09)	M27	DMU_C(14)	N30	DMU_C(20)
L25	DMU_C(08)	M28	3.3V	N31	DMU_C(19)
L26	DMU_C(07)	M29	DMU_C(13)	N32	DMU_C(18)
L27	DMU_C(06)	M30	GND	N33	DMU_C(17)
L28	DMU_C(05)	M31	DMU_C(12)	P01	$\overline{\text{DASL\_Out\_B(0)}}$
L29	DMU_C(04)	M32	V_{DD}	P02	GND
L30	DMU_C(03)	M33	DMU_C(11)	P03	DASL_Out_B(3)
L31	DMU_C(02)	N01	DASL_Out_B(1)	P04	V_{DD}
L32	DMU_C(01)	N02	DASL_Out_B(2)	P05	Unused
L33	DMU_C(00)	N03	$\overline{\text{DASL\_Out\_B(3)}}$	P06	GND
P07	Unused	R10	Unused	T13	Unused
P08	2.5V	R11	Unused	T14	V_{DD}
P09	Unused	R12	Unused	T15	LU_Data(24)
P10	GND	R13	Unused	T16	GND
P11	Unused	R14	Unused	T17	V_{DD}
P12	2.5V	R15	GND	T18	GND
P13	Unused	R16	Unused	T19	Send_Grant_B

Table 33: Complete Signal Pin Listing by Grid Position (Page 9 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
P14	GND	R17	D4_Data(14)	T20	V_{DD}
P15	D4_Data(28)	R18	DS1_DQS(3)	T21	RES_Data
P16	V_{DD}	R19	GND	T22	GND
P17	GND	R20	MC_Grant_B(0)	T23	DMU_B(19)
P18	V_{DD}	R21	Switch_BNA	T24	3.3V
P19	DS0_Data(00)	R22	DMU_B(18)	T25	$\overline{\text{JTAG\_TRst}}$
P20	GND	R23	DMU_B(13)	T26	GND
P21	MC_Grant_B(1)	R24	DMU_B(12)	T27	DMU_B(17)
P22	3.3V	R25	DMU_B(11)	T28	V_{DD}
P23	DMU_B(02)	R26	DMU_B(10)	T29	DMU_B(16)
P24	GND	R27	DMU_B(09)	T30	GND
P25	DMU_B(01)	R28	DMU_B(08)	T31	DMU_B(15)
P26	V_{DD}	R29	DMU_B(07)	T32	3.3V
P27	DMU_B(00)	R30	DMU_B(06)	T33	DMU_B(14)
P28	GND	R31	DMU_B(05)	U01	LU_Data(30)
P29	DMU_C(30)	R32	DMU_B(04)	U02	Thermal_Out
P30	3.3V	R33	DMU_B(03)	U03	LU_Data(35)
P31	DMU_C(29)	T01	Spare_Tst_Rcvr(3)	U04	Thermal_In
P32	GND	T02	V_{DD}	U05	Spare_Tst_Rcvr(0)
P33	DMU_C(28)	T03	Spare_Tst_Rcvr(8)	U06	Testmode(1)
R01	Unused	T04	GND	U07	Unused
R02	DASL_Out_B(0)	T05	Unused	U08	LU_Data(08)
R03	$\overline{\text{DASL\_Out\_B(1)}}$	T06	2.5V	U09	Unused
R04	LU_Addr(08)	T07	LU_Data(03)	U10	LU_Data(34)
R05	LU_Data(33)	T08	GND	U11	Unused
R06	Unused	T09	LU_Data(02)	U12	LU_Data(11)
R07	LU_Data(04)	T10	V_{DD}	U13	LU_Data(01)
R08	LU_Data(05)	T11	Unused	U14	GND
R09	Unused	T12	GND	U15	LU_Data(00)
U16	V_{DD}	V19	MGrant_A(0)	W22	JTAG_TDI
U17	GND	V20	V_{DD}	W23	DMU_A(13)
U18	V_{DD}	V21	I_FreeQ_Th	W24	DMU_A(12)
U19	MGrant_A(1)	V22	GND	W25	DMU_A(11)
U20	GND	V23	Rx_LByte(1)	W26	DMU_A(10)
U21	RES_Sync	V24	V_{DD}	W27	DMU_A(09)
U22	JTAG_TMS	V25	DMU_B(20)	W28	DMU_A(08)
U23	Rx_LByte(0)	V26	GND	W29	DMU_A(07)

Table 33: Complete Signal Pin Listing by Grid Position (Page 10 of 10)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
U24	DMU_B(29)	V27	DMU_A(02)	W30	DMU_A(06)
U25	DMU_B(28)	V28	3.3V	W31	DMU_A(05)
U26	DMU_B(27)	V29	DMU_A(01)	W32	DMU_A(04)
U27	DMU_B(26)	V30	GND	W33	DMU_A(03)
U28	DMU_B(25)	V31	DMU_A(00)	Y01	DASL_Out_A(0)
U29	DMU_B(24)	V32	V_{DD}	Y02	GND
U30	DMU_B(23)	V33	DMU_B(30)	Y03	$\overline{\text{DASL\_In\_A(7)}}$
U31	DMU_B(22)	W01	LU_Data(20)	Y04	2.5V
U32	DMU_B(21)	W02	$\overline{\text{DASL\_Out\_A(0)}}$	Y05	LU_Data(32)
U33	Spare_Tst_Rcvr(9)	W03	DASL_Out_A(1)	Y06	GND
V01	Spare_Tst_Rcvr(6)	W04	LU_Data(13)	Y07	LU_Data(17)
V02	2.5V	W05	LU_Data(07)	Y08	V_{DD}
V03	Spare_Tst_Rcvr(7)	W06	LU_Data(06)	Y09	LU_Data(22)
V04	GND	W07	LU_Data(15)	Y10	GND
V05	Testmode(0)	W08	LU_Data(16)	Y11	LU_Addr(01)
V06	V_{DD}	W09	LU_Data(21)	Y12	2.5V
V07	LU_Data(09)	W10	LU_Data(25)	Y13	VRef2(6)
V08	GND	W11	LU_Data(31)	Y14	GND
V09	LU_Data(10)	W12	LU_Data(26)	Y15	D1_Data(05)
V10	2.5V	W13	LU_Data(19)	Y16	V_{DD}
V11	LU_Data(14)	W14	LU_Data(27)	Y17	GND
V12	GND	W15	GND	Y18	V_{DD}
V13	LU_Data(18)	W16	D1_Addr(10)	Y19	D2_Data(15)
V14	V_{DD}	W17	D3_Data(08)	Y20	GND
V15	LU_Data(12)	W18	D2_Data(02)	Y21	MC_Grant_A(0)
V16	GND	W19	GND	Y22	3.3V
V17	V_{DD}	W20	Send_Grant_A	Y23	DMU_A(19)
V18	GND	W21	MC_Grant_A(1)	Y24	GND
Y25	DMU_A(18)	Y28	GND	Y31	DMU_A(15)
Y26	3.3V	Y29	DMU_A(16)	Y32	GND
Y27	DMU_A(17)	Y30	V_{DD}	Y33	DMU_A(14)

2.4 IEEE 1149 (JTAG) Compliance

2.4.1 Statement of JTAG Compliance

The NP4GS3 is compliant with IEEE Standard 1149.1a.

2.4.2 JTAG Compliance Mode

Compliance with IEEE 1149.1a is enabled by applying a compliance-enable pattern to the compliance-enable inputs as shown in *Table 34*.

Table 34: JTAG Compliance-Enable Inputs

Compliance-Enable Inputs	Compliance-Enable Pattern
Testmode(1:0)	'10'
Spare_Tst_Rcvr(9)	1
Spare_Tst_Rcvr(4)	1
Spare_Tst_Rcvr(3)	1
Spare_Tst_Rcvr(2)	1

Note: To achieve reset of the JTAG test logic, the JTAG_TRst input must be driven low when the compliance-enable pattern is applied.

2.4.3 JTAG Implementation Specifics

All mandatory JTAG public instructions are implemented in the NP4GS3's design. *Table 35* documents all implemented public instructions.

Table 35: Implemented JTAG Public Instructions

Instruction Name	Binary Code	[1] Serial Data Reg connected to TDI/TDO	I/O Control Source (driver data and driver enable)
BYPASS	111 1111	Bypass	System, functional values
CLAMP	111 1101	Bypass	JTAG
EXTEST	111 1000	BoundaryScan	JTAG
HIGHZ	111 1010	Bypass	JTAG, all drivers disabled
SAMPLE/PRELOAD	111 1001	BoundaryScan	System, functional values

1. Chip TDO output driver is only enabled during TAP Controller states Shift_IR and Shift_DR.

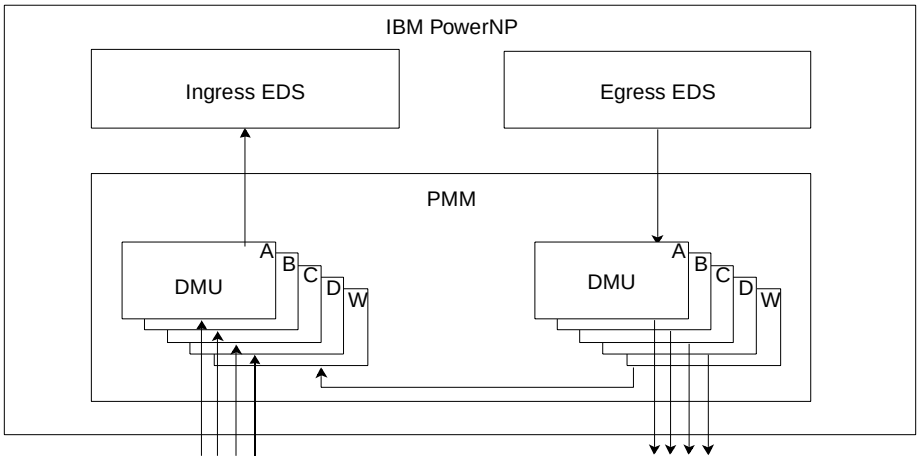
2.4.4 Brief Overview of JTAG Instructions

Instruction	Description
BYPASS	Connects the bypass data register between the TDI and TDO pins. The bypass data register is a single shift-register stage that provides a minimum-length serial path between the TDI and TDO pins when no test operation of the device is required. Bypass does not disturb the normal functional connection and control of the I/O pins.
CLAMP	Causes all output pins to be driven from the corresponding JTAG parallel boundary scan register. The SAMPLE/PRELOAD instruction is typically used to load the desired values into the parallel boundary scan register. Since the clamp instruction causes the serial TDI/TDO path to be connected to the bypass data registers, scanning through the device is very fast when the clamp instruction is loaded.
EXTEST	Allows the JTAG logic to control output pin values by connecting each output data and enable signal to its corresponding parallel boundary scan register. The desired controlling values for the output data and enable signals are shifted into the scan boundary scan register during the shiftDR state. The parallel boundary scan register is loaded from the scan boundary scan register during the updateDR state. The EXTEST instruction also allows the JTAG logic to sample input receiver and output enable values. The values are loaded into the scan boundary scan register during captureDR state.
HIGHZ	Causes the JTAG logic to tri-state all output drivers while connecting the bypass register in the serial TDI/TDO path.
SAMPLE/PRELOAD	Allows the JTAG logic to sample input pin values and load the parallel boundary scan register without disturbing the normal functional connection and control of the I/O pins. The sample phase of the instruction occurs in captureDR state, at which time the scan boundary scan register is loaded with the corresponding input receiver and output enable values. (Note that for input pins that are connected to a common I/O, the scan boundary scan register only updates with a input receiver sample if the corresponding output driver of the common I/O is disabled; otherwise the scan register is updated with the output data signal. The desired controlling values for the output pins are shifted into the scan boundary scan register during the shiftDR state and loaded from the scan boundary scan register to the parallel boundary scan register during the updateDR state.

3. PMM Overview

The Physical MAC Multiplexer (PMM) Unit interfaces with the network processor's external ports in the ingress (I-PMM) and egress (E-PMM) directions. The PMM includes five Data Mover Units (DMUs A, B, C, D, and the Wrap DMU). Four DMUs (A, B, C, and D) can each be independently configured as an Ethernet MAC or a POS MAC. A complete set of performance statistics is kept on a per port basis in either mode. Each DMU's data throughput capability is 1 Gbps in both the ingress and the egress directions. The Wrap DMU is used as a wrap path to enable traffic generated by the NP4GS3's egress side to be sent up through the switch fabric through the ingress side.

Figure 21: PMM Overview



3.1 Ethernet Overview

A DMU configured in Ethernet mode can support either one port of Gigabit Ethernet or ten ports of 10/100 Ethernet. *Figure 25: Ethernet Mode* on page 95 shows an NP4GS3 with DMU-A and DMU-B configured as Gigabit Ethernet MACs. When in Gigabit mode, each DMU can be configured with either a Gigabit Media Independent Interface (GMII) or a TBI interface. DMU-C and DMU-D are shown configured in 10/100 Megabit Ethernet Serial Media Independent Interface (SMII) mode. When in the SMII mode, the single DMU MAC time division multiplexes the ten ports. Each of these four DMUs can be configured in any of the Ethernet operation modes.

Timing diagrams for the SMII, GMII, and TBI Ethernet interfaces are shown below.

Figure 22: SMII Timing Diagram

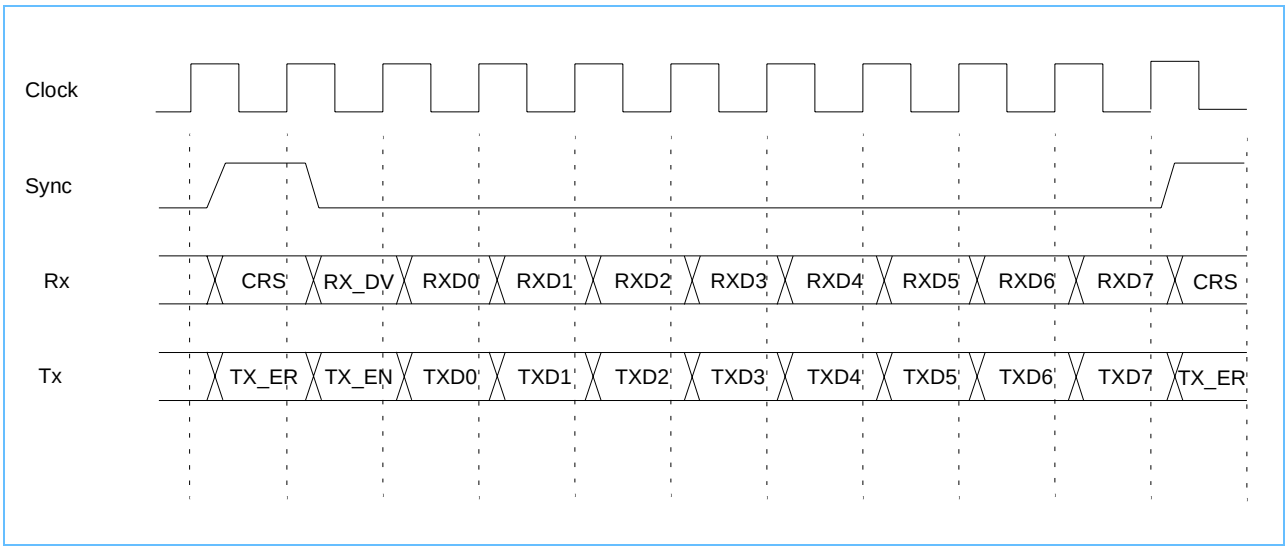


Figure 23: GMII Timing Diagrams

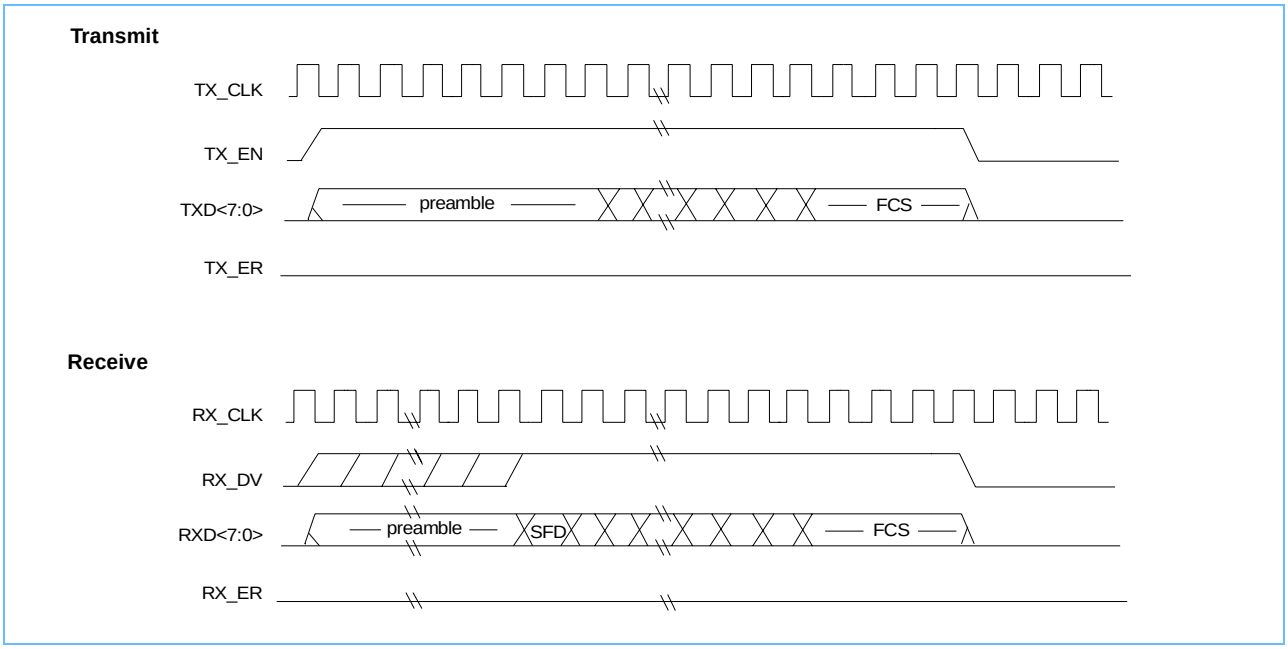


Figure 24: TBI Timing Diagrams

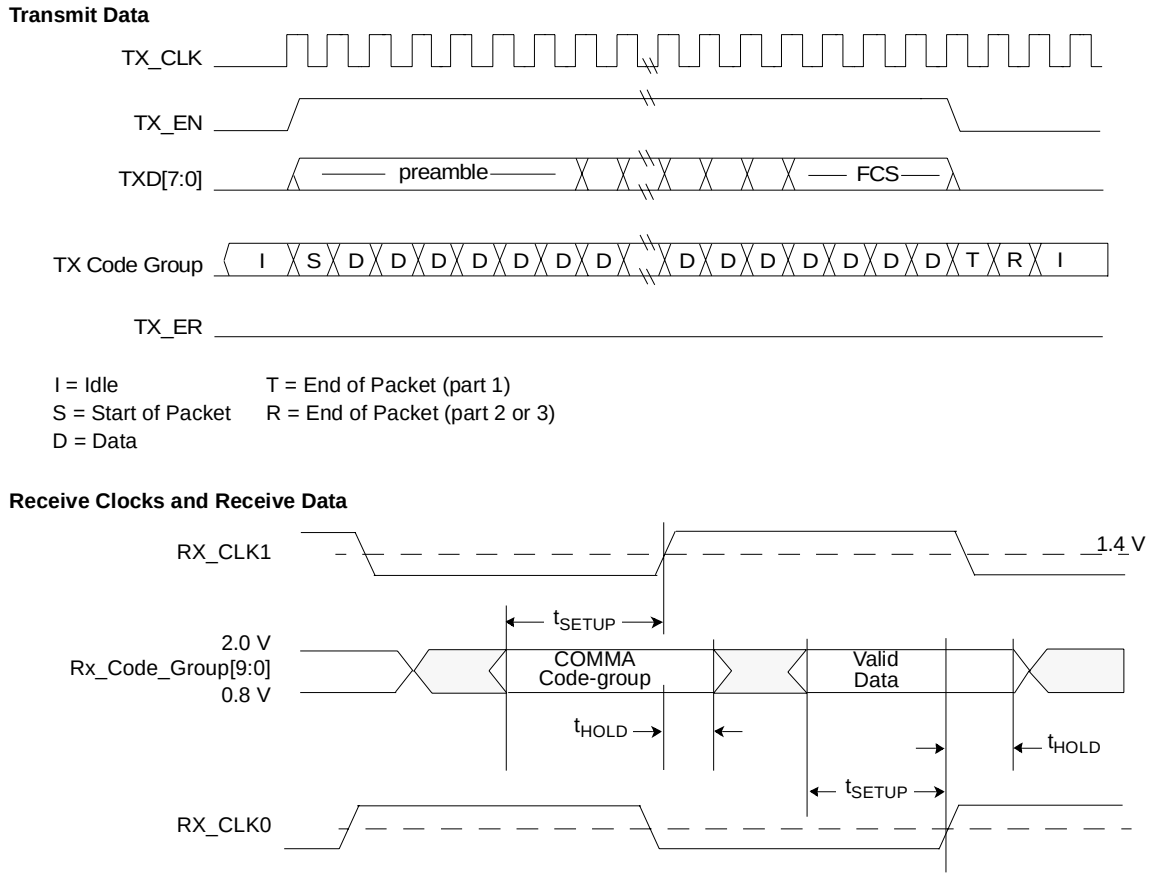


Figure 25: Ethernet Mode

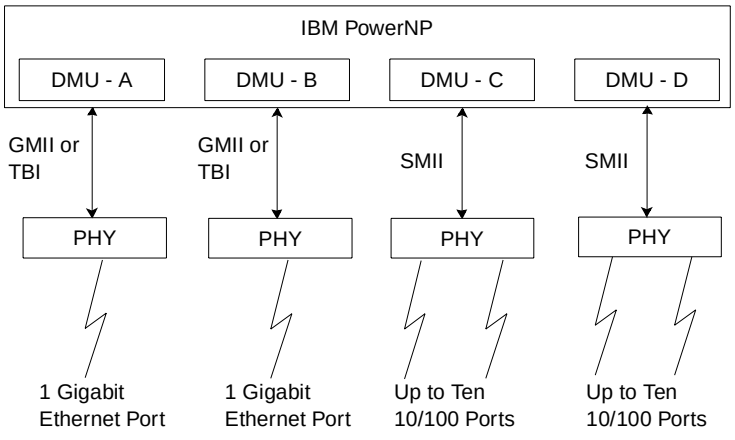


Figure 26: GMII POS Mode Timing Diagram

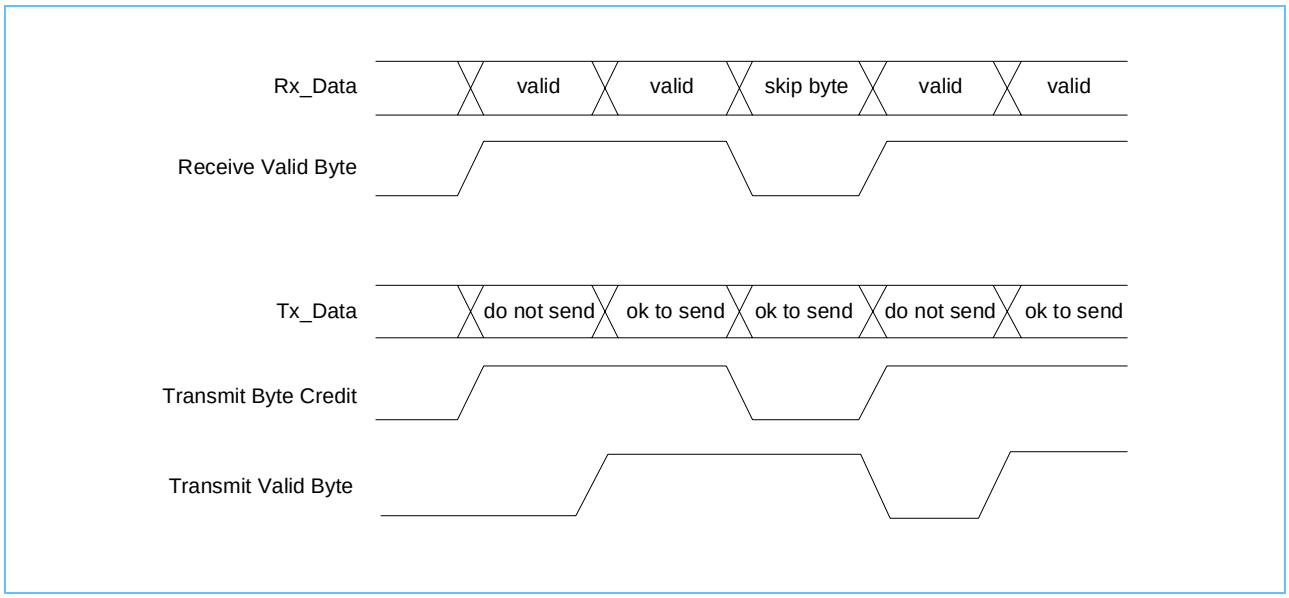


Figure 25: Ethernet Mode on page 95 shows an example of Ethernet configuration of the DMUs. Any of the four DMUs can be configured in Gigabit Ethernet or Ten 10/100 ports.

Table 36: Ingress Ethernet Counters on page 97 and *Table 37: Egress Ethernet Counters* on page 99 show the statistics counters kept in each DMU when it operates in Ethernet Mode. These counters are accessible through the CAB interface. (For information on CAB addresses, refer to the IBM PowerNP NP4GS3 Hardware Reference Manual, sections 3 and 6.)

Table 36: Ingress Ethernet Counters

Name / Group	Counter No.	Group	Description	Notes
Short Frames	x'00'	4	Total number of frames received that were less than 64 octets long and were otherwise well formed (had a good CRC).	1, 2
Fragments	x'01'		Total number of frames received that were less than 64 octets long and had a bad CRC.	1, 2
Frames Received (64 octets)	x'02'		Total number of frames (including bad frames) received that were 64 octets in length.	1, 2
Frames Received (65 to 127 octets)	x'03'		Total number of frames (including bad frames) received that were between 65 and 127 octets in length inclusive.	1, 2
Frames Received (128 to 255 octets)	x'04'		Total number of frames (including bad frames) received that were between 128 and 255 octets in length inclusive.	1, 2
Frames Received (256 to 511 octets)	x'05'		Total number of frames (including bad frames) received that were between 256 and 511 octets in length inclusive.	1, 2
Frames Received (512 to 1023 octets)	x'06'		Total number of frames (including bad frames) received that were between 512 and 1023 octets in length inclusive.	1, 2
Frames Received (1024 to 1518 octets)	x'07'		Total number of frames (including bad frames) received that were between 1024 and 1518 octets in length inclusive.	1, 2
Jumbo Frames	x'14'		Total number of frames (including bad frames) received with a length between 1519 and 9018 octets, or between 1519 and 9022 octets if VLAN is asserted. If Jumbo is not asserted the frame is a long frame.	
Long Frames Received	x'08'		Total number of long frames received with a good CRC that were either: 1) Longer than 1518 octets (excluding framing bits, but including CRC octets), and were otherwise well formed (good CRC), VLAN and Jumbo deasserted. 2) VLAN frames that were longer than 1522 octets, with Jumbo deasserted, 3) Jumbo frames that were longer than 9018 octets, with VLAN deasserted 4) Jumbo VLAN frames that were longer than 9022 octets.	3
Jabber	x'09'		Total number of long frames received with bad CRC that were either: 1) Longer than 1518 octets (excluding framing bits, but including CRC octets), and were not well formed (Bad CRC), VLAN and Jumbo not asserted, 2) VLAN frames that were longer than 1522 octets, with Jumbo deasserted, 3) Jumbo frames that were longer than 9018 octets, with VLAN deasserted 4) Jumbo VLAN frames that were longer than 9022 octets.	3
Frames with Bad CRC	x'0A'	3	Total number of frames received that were not long frames, but had bad CRC.	2
Unicast Frames Received	x'0B'		Total number of good frames received that were directed to the unicast address (excluding multicast frames, broadcast frames, or long frames).	
Broadcast Frames Received	x'0C'		Total number of good frames received that were directed to the broadcast address (excluding multicast frames or long frames).	
Multicast Frames Received	x'0D'		Total number of good frames received that were directed to the multicast address (excluding broadcast frames, or long frames).	

1. The states of VLAN or Jumbo have no effect on this count.
2. Excluding framing bits but including CRC octets
3. Reception of frames meeting the criteria for this counter are aborted by the hardware. Abort is indicated in the Ingress Port Control Block and in the Ingress Frame Control Block. If the frame has not been forwarded by the picocode, the picocode must enqueue the frame to the ingress discard queue. If the frame has been forwarded, then the egress hardware will discard the frame. At the MAC further reception is inhibited until the next frame starts.

Table 36: Ingress Ethernet Counters (Continued)

Name / Group	Counter No.	Group	Description	Notes
Total Frames Received	x'0E'	2	Total number of frames (including bad frames, broadcast frames, multicast frames, unicast frames, and long frames) received.	
Receive Errors	x'0F'		Total number of frames received in which the PHY detected an error and asserted the Rx_Err signal.	
Overruns	x'13'		Total number of frames received when the PMM internal buffer was full and the frame couldn't be stored. Includes frames in the process of being received when the PMM internal buffer becomes full.	
Pause Frames	x'10'		Total number of MAC pause frames received that were well formed and had a good CRC.	
Total Pause Time	x'11'	1	Total amount of time spent in a pause condition as a result of receiving a good pause MAC frame.	
Total Octets Received	x'12'	0	Total number of octets of data (including those in bad frames) received on the network.	2

1. The states of VLAN or Jumbo have no effect on this count.
2. Excluding framing bits but including CRC octets
3. Reception of frames meeting the criteria for this counter are aborted by the hardware. Abort is indicated in the Ingress Port Control Block and in the Ingress Frame Control Block. If the frame has not been forwarded by the picocode, the picocode must enqueue the frame to the ingress discard queue. If the frame has been forwarded, then the egress hardware will discard the frame. At the MAC further reception is inhibited until the next frame starts.

Table 37: Egress Ethernet Counters

Name	Counter No.	Group	Description	Cnt
Short Frames	x'00'	4	Total number of frames transmitted with less than 64 octets that had good CRC.	1, 2
Runt Frames (Bad CRC)	x'01'		Total number of frames transmitted with less than 64 octets that had bad CRC.	1, 2
Frames Transmitted (64 octets)	x'02'		Total number of frames (including bad frames) transmitted that were 64 octets in length.	1
Frames Transmitted (65 to 127 octets)	x'03'		Total number of frames (including bad frames) transmitted that were between 65 and 127 octets in length inclusive.	1, 3
Frames Transmitted (128 to 255 octets)	x'04'		Total number of frames (including bad frames) transmitted that were between 128 and 255 octets in length inclusive	1, 3
Frames Transmitted (256 to 511 octets)	x'05'		Total number of frames (including bad frames) transmitted that were between 256 and 511 octets in length inclusive.	1, 3
Frames Transmitted (512 to 1023 octets)	x'06'		Total number of frames (including bad frames) transmitted that were between 512 and 1023 octets in length inclusive.	1, 3
Frames Transmitted (1024 to 1518 octets)	x'07'		Total number of frames (including bad frames) transmitted that were between 1024 and 1518 octets in length inclusive.	1, 3
Jumbo Frames	x'16'		Total number of frames (including bad frames) transmitted with a length between 1519 and 9018 octets, or between 1523 and 9022 if VLAN is asserted. If Jumbo is not asserted, the frame is a long frame.	
Long Frames Transmitted	x'08'		Total number of frames with good CRC transmitted that were either: 1) Longer than 1518 octets (excluding framing bits, but including CRC octets), and were otherwise well formed (good CRC), VLAN and Jumbo are deasserted. 2) VLAN frames that were longer than 1522 octets with Jumbo deasserted, 3) Jumbo frames that were longer than 9018 octets with Jumbo deasserted 4) Jumbo VLAN frames that were longer than 9022 octets.	
Jabber	x'09'		Total number of frames with bad CRC transmitted that were either: 1) Longer than 1518 octets (excluding framing bits, but including CRC octets), and were otherwise well formed (good CRC), VLAN and Jumbo are deasserted. 2) VLAN frames that were longer than 1522 octets with Jumbo deasserted, 3) Jumbo frames that were longer than 9018 octets with Jumbo deasserted 4) Jumbo VLAN frames that were longer than 9022 octets.	
Late Collisions	x'0A'		Total number of frames transmitted that experienced a network collision after 64 bytes of the frame had been transmitted.	1

1. The states of VLAN or Jumbo have no effect on this count.
2. Including the frame type byte.
3. Excluding framing bits but including CRC octets.

Table 37: Egress Ethernet Counters (Continued)

Name	Counter No.	Group	Description	Cnt
Total Collisions	x‘0B’	3	Best estimate of the total number of collisions on this Ethernet segment.	1
Single Collisions	x‘0C’		Total number of frames transmitted that experienced one collision before 64 bytes of the frame were transmitted on the network.	1
Multiple Collisions	x‘0D’		Total number of frames transmitted that experienced more than one collision before 64 bytes of the frame were transmitted on the network.	1
Excessive Deferrals	x‘0E’		Total number of frames whose transmission could not be started before the deferral time out expired. The deferral time out value is 24,416 media bit times.	1
Underruns	x‘0F’		Total number of frames that were not completely transmitted because the data could not be obtained from the Egress EDS fast enough to maintain the media data rate.	
Aborted Frames	x‘17’		Total number of frames that had link status pointer parity errors and were aborted by the Egress PMM.	
CRC Error	x‘10’	2	Total number of frames transmitted that had a legal length (excluding framing bit, but including CRC octets) of between 64 and 9018 octets, (64 and 9022 for VLAN frames) inclusive, but had a bad CRC.	
Excessive Collisions	x‘11’		Total number of frames that experienced more than 16 collisions during transmit attempts. These frames are dropped and not transmitted.	
Unicast Frames Transmitted	x‘12’		Total number of good frames transmitted that were directed to the unicast address (not including multicast frames or broadcast frames).	
Broadcast Frames Transmitted	x‘13’		Total number of good frames transmitted that were directed to the broadcast address (not including multicast frames).	
Multicast Frames Transmitted	x‘14’		Total number of good frames transmitted that were directed to the multicast address (not including broadcast frames).	
Total Octets Transmitted	x‘15’	0	Total number of octets of data (including those in bad frames) transmitted on the network.	3

1. The states of VLAN or Jumbo have no effect on this count.
2. Including the frame type byte.
3. Excluding framing bits but including CRC octets.

Table 38: Ethernet Support

Feature		Ethernet Mode		
		10/100 MAC	1000 Mbps MAC	1000 Mbps Physical Coding Sublayer
Fully compatible with IEEE standard	802.3: 1993 (E)	X		
	802.3u / D5.3	X		
	802.3z/D4 standards		X	
	802.3 Clause 36 (TBI) standards			X
Compliant with RFC 1757 Management Registers and Counters (Tx/Rx Counters) Additional receive counters for total number of pause frames and total aggregate pause time Additional transmit counters for number of single collisions and number of multiple collisions		X	X	
Supports the IEEE standards on flow control by honoring received pause frames and inhibiting frame transmission while maintaining pause counter statistics		X	X	
Supports the Serial Medium Independent Interface (SMII) to the PHY Interfaces with up to ten PHYs that support the SMII interface. Each of the ten interfaces can have a bit rate of either 10 Mbps or 100 Mbps		X		
Capable of handling ten ports of 10 Mbps or 100 Mbps media speeds, any speed mix		X		
Supports half duplex operations at media speed on all ports		X		
Supports Binary Exponential Back-off (BEB) compliant with the IEEE Std. 802.3: 1993 (E)		X		
Supports full duplex point-to-point operations at media speed.		X	X	
Detects VLAN (8100 frame type) Ethernet frames and accounts for them when calculating long frames		X	X	
Supports two Ethernet frame types (programmable) and, based on these, detects received frames with a type field that matches one of those types. A match instructs the Multi-port 10/100 MAC to strip the DA, SA, and Type fields from the received frame and to instruct the higher chip functions to queue the frame in a different queue. An example of a special function is to identify Ethernet encapsulated guided frames. A mismatch results in normal frame queueing and normal higher chip processing.		X	X	
Programmable cyclic redundancy check (CRC) Insertion on a frame basis With CRC Insertion disabled, MAC transmits frames as is (suitable for switch environments) With CRC Insertion enabled, the MAC calculates and inserts the CRC			X	
Includes jumbo frame support. When configured, can transmit and receive up to 9018-byte non-VLAN frames or up to 9022-byte VLAN frames		X	X	
Transfers received data to the upper chip layers using a proprietary 16-byte wide interface. In the transmit direction, data from the data mover is sent to the MAC on a single 8-bit bus.		X	X	
Supports the Gigabit Medium Independent Interface (GMII) to the physical TBI layer			X	
Supports the IBM TBI “Valid Byte” signal			X	
Can be combined with the TBI to form a complete TBI solution			X	
Interfaces with any PMA/PMI physical layer using the PMA service interface defined in the IEEE 802.3 standard				X
Synchronizes the data received from the PMA (two phase) clock with the MAC (single phase) clock. Provides a signal to the MAC indicating those clock cycles that contain new data.				X
Checks the received code groups (10 bits) for commas and establishes word synchronization				X
Calculates and checks the TBI running disparity				X
Supports auto-negotiation including two next pages				X
Interfaces with the 1000 Mbps Ethernet MAC through the GMII to form a complete TBI solution				X

3.2 POS Overview

Each of the four 8-bit DMUs supports Packet Over SONET (POS) and allows connection to OC-3c, OC-12, or OC-12c framers. To provide an OC-48 link, all four DMUs can be attached to a single framer, with four different 8 bit OC-12c channels, or to four different OC-12c framers. To provide an OC-48c link, DMU-A can be configured to attach to a 32-bit framer and the other three DMUs, although configured for OC-48 mode, are disabled, except for their interface pins.

Figure 32 is an example of an OC-3c/12/12c configuration, in which each of the four DMUs can be configured to operate in either a single port mode or in a multi-port mode servicing four ports. Each of the four DMUs supports an 8-bit data interface in both the ingress and egress directions.

3.2.1 POS Counters

The following tables provide information about the counters maintained to support POS interfaces.

3.2.1.1 Long Frames

A long frame is defined as a packet whose byte count exceeds the value held in the Packet Over SONET Maximum Frame Size (POS_Max_FS) register (see *IBM PowerNP Configuration* on page 339), the byte count includes the CRC octets.

Reception of packets meeting the criteria for long frames are aborted by the hardware. Abort is indicated in the Ingress Port Control Block and in the Ingress Frame Control Block. If the packet has not been forwarded by the picocode, the picocode must enqueue the packet to the ingress discard queue. If the packet has been forwarded, then the egress hardware will discard the frame. At the MAC, further reception is inhibited until the next packet starts.

Table 39: Receive Counter RAM Addresses for Ingress POS MAC

Port	Name	Counter Number	Description
Port 0	Long Frames Received	x‘00’	Total number of long frames received with a good CRC that were longer than the value contained in the POS Maximum Frame Size Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘01’	Total number of frames received that had a length of the value contained in the POS Maximum Frame Size Register (POS_Max_FS) or less, but had a bad CRC.
	Total Good Frames Received	x‘02’	Total number of frames (excluding frames with bad CRC, and long frames) received.
	Receive Errors	x‘03’	Total number of frames received in which the Framer detected an error and asserted the Rx_Err signal.
	Total Octets Received (including frames with errors)	x‘10’	Total number of octets of data (including those in bad frames) received on the network.

Table 39: Receive Counter RAM Addresses for Ingress POS MAC (Continued)

Port	Name	Counter Number	Description
Port 1	Long Frames Received	x‘04’	Total number of long frames received with a good CRC that were longer than the value contained in the POS Maximum Frame Size Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘05’	Total number of frames received that had a length of the value contained in the POS Maximum frame Size Register (POS_Max_FS) or less, but had a bad CRC.
	Total Good Frames Received	x‘06’	Total number of frames (excluding frames with a bad CRC, and long frames) received.
	Receive Errors	x‘07’	Total number of frames received in which the Framer detected an error and asserted the Rx_Err signal.
	Total Octets Received (including frames with errors)	x‘11’	Total number of octets of data (including those in bad frames) received on the network.
Port 2	Long Frames Received	x‘08’	Total number of long frames received with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘09’	Total number of frames received that had a length of the value contained in the POS Maximum Frame Size Register (POS_Max_FS) or less, but had a bad CRC.
	Total Good Frames Received	x‘0A’	Total number of frames (excluding frames with a bad CRC, and long frames) received.
	Receive Errors	x‘0B’	Total number of frames received in which the Framer detected an error and asserted the Rxerr signal.
	Total Octets Received (including frames with errors)	x‘12’	Total number of octets of data (including those in bad frames) received on the network.
Port 3	Long Frames Received	x‘0C’	Total number of long frames received with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘0D’	Total number of frames received that had a length of the value contained in the POS Maximum Frame Size Register (POS_Max_FS) or less, but had a bad CRC.
	Total Good Frames Received	x‘0E’	Total number of frames (excluding frames with a bad CRC, and long frames) received.
	Receive Errors	x‘0F’	Total number of frames received in which the Framer detected an error and asserted the Rx_Err signal.
	Total Octets Received (including frames with errors)	x‘13’	Total number of octets of data (including those in bad frames) received on the network.

Table 40: Transmit Counter RAM Addresses for Egress POS MAC

Port	Name	Counter Number	Description
Port 0	Long Frames Transmitted	x‘00’	Total number of long frames transmitted with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘01’	Total number of frames transmitted that had a length of the value contained in the POS Maximum Frame Register (POS_Max_FS) or less, but had bad CRC.
	Total Good Frames Transmitted	x‘02’	Total number of frames (excluding frames with a bad CRC, and long frames) transmitted.
	Transmit Underruns	x‘03’	Total number of frames attempted to be transmitted but an under-run occurred in the NP4GS3.
	Total Octets Transmitted (including frames with errors)	x‘10’	Total number of octets of data (including those in bad frames) transmitted on the network.
	Aborted Frames	x‘14’	Total number of frames that had link status pointer parity errors and were aborted by the Egress PMM.
Port 1	Long Frames Received	x‘04’	Total number of long frames transmitted with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘05’	Total number of frames transmitted that had a length of the value contained in the POS Maximum Frame Register (POS_Max_FS) or less, but had bad CRC.
	Total Good Frames Transmitted	x‘06’	Total number of frames (excluding frames with a bad CRC, and long frames) transmitted.
	Transmit Underruns	x‘07’	Total number of frames attempted to be transmitted but an under-run occurred in the NP4GS3.
	Total Octets Transmitted (including frames with errors)	x‘11’	Total number of octets of data (including those in bad frames) transmitted on the network.
	Aborted Frames	x‘15’	Total number of frames that had link status pointer parity errors and were aborted by the Egress PMM.
Port 2	Long Frames Transmitted	x‘08’	Total number of long frames transmitted with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x‘09’	Total number of frames transmitted that had a length of the value contained in the POS Maximum Frame Register (POS_Max_FS) or less, but had bad CRC.
	Total Good Frames Transmitted	x‘0A’	Total number of frames (excluding frames with a bad CRC, and long frames) transmitted.
	Transmit Underruns	x‘0B’	Total number of frames attempted to be transmitted but an under-run occurred in the NP4GS3.
	Total Octets Transmitted (including frames with errors)	x‘12’	Total number of octets of data (including those in bad frames) transmitted on the network.
	Aborted Frames	x‘16’	Total number of frames that had link status pointer parity errors and were aborted by the Egress PMM.

Table 40: Transmit Counter RAM Addresses for Egress POS MAC (Continued)

Port	Name	Counter Number	Description
Port 3	Long Frames Transmitted	x'0C'	Total number of long frames transmitted with a good CRC that were longer than the value contained in the POS Maximum Frame Register (POS_Max_FS) including CRC octets.
	Frames with Bad CRC	x'0D'	Total number of frames transmitted that had a length of the value contained in the POS Maximum Frame Register (POS_Max_FS) or less, but had bad CRC.
	Total Good Frames Transmitted	x'0E'	Total number of frames (excluding frames with a bad CRC, and long frames) transmitted.
	Transmit Underruns	x'0F'	Total number of frames attempted to be transmitted but an under-run occurred in the NP4GS3.
	Total Octets Transmitted (including frames with errors)	x'13'	Total number of octets of data (including those in bad frames) transmitted on the network.
	Aborted Frames	x'17'	Total number of frames that had link status pointer parity errors and were aborted by the Egress PMM.

Figure 27: Receive POS8 Interface Timing for 8-bit data bus (OC-3c, OC-12, OC-12c, and OC-48).

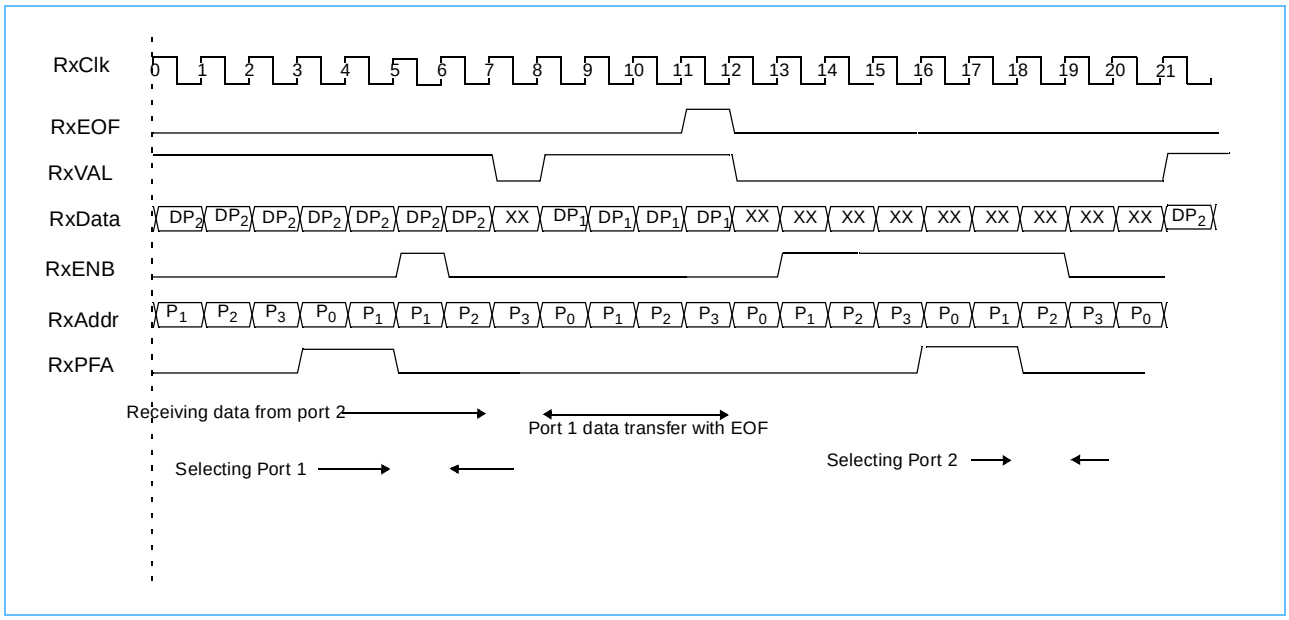


Figure 28: Receive POS32 Interface Timing for 32-bit data bus (OC-48c).

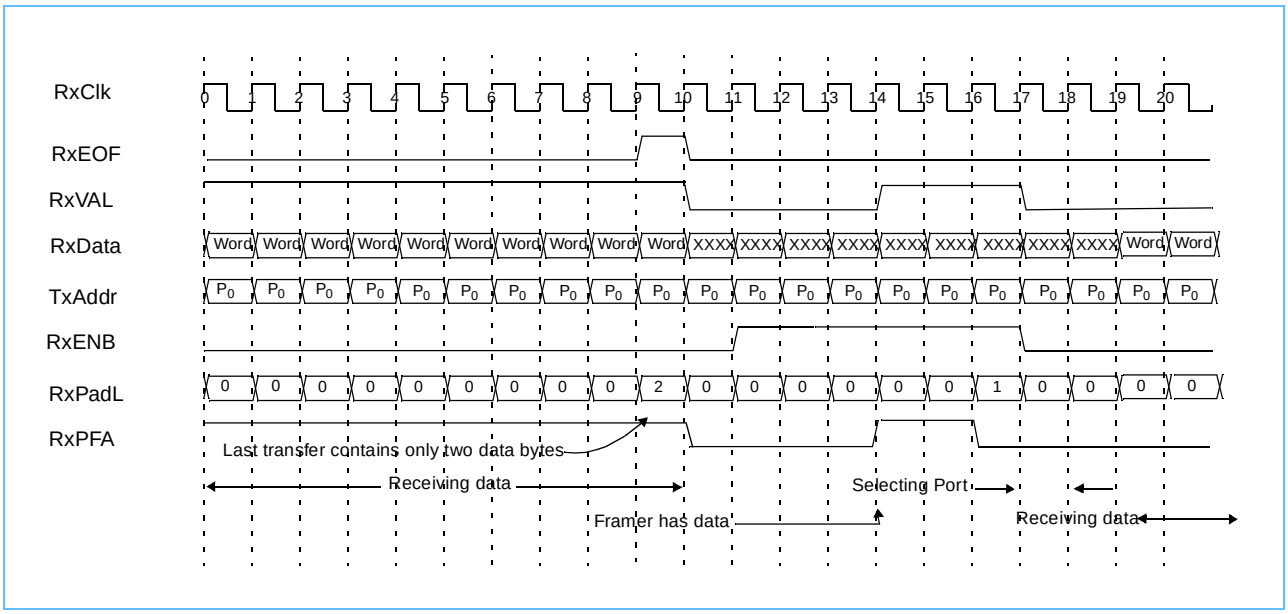


Figure 29: Transmit POS8 Interface Timing for 8-bit data bus (OC-3c, OC-12)

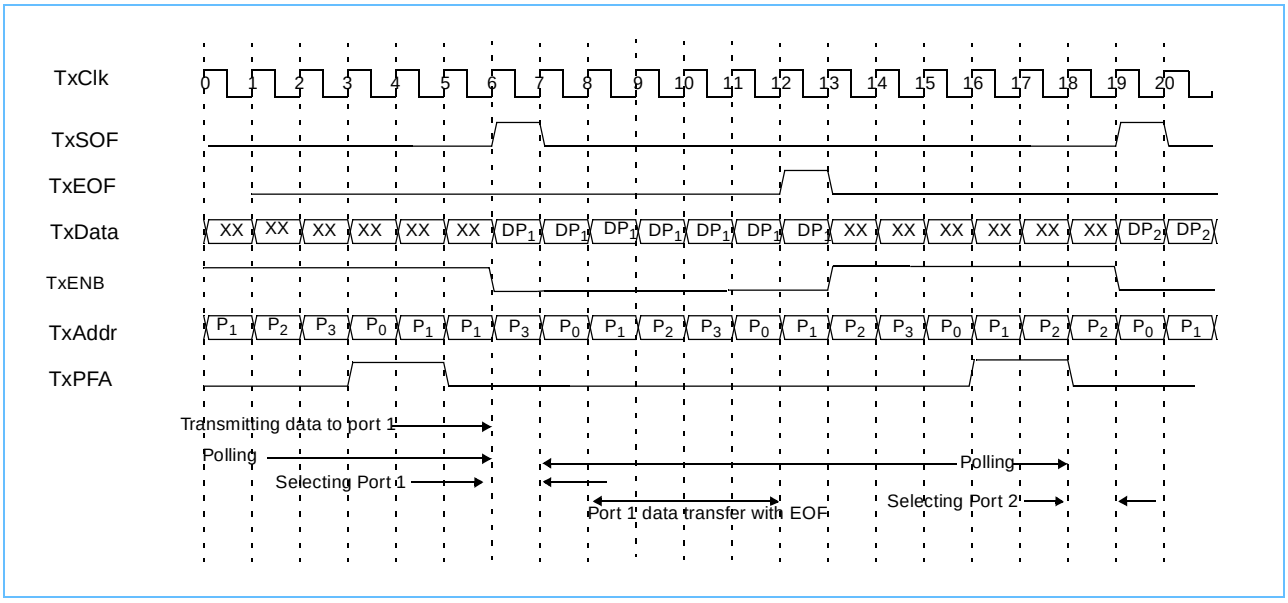


Figure 30: Transmit POS8 Interface Timing for 8-bit data bus (OC-12c, OC-48)

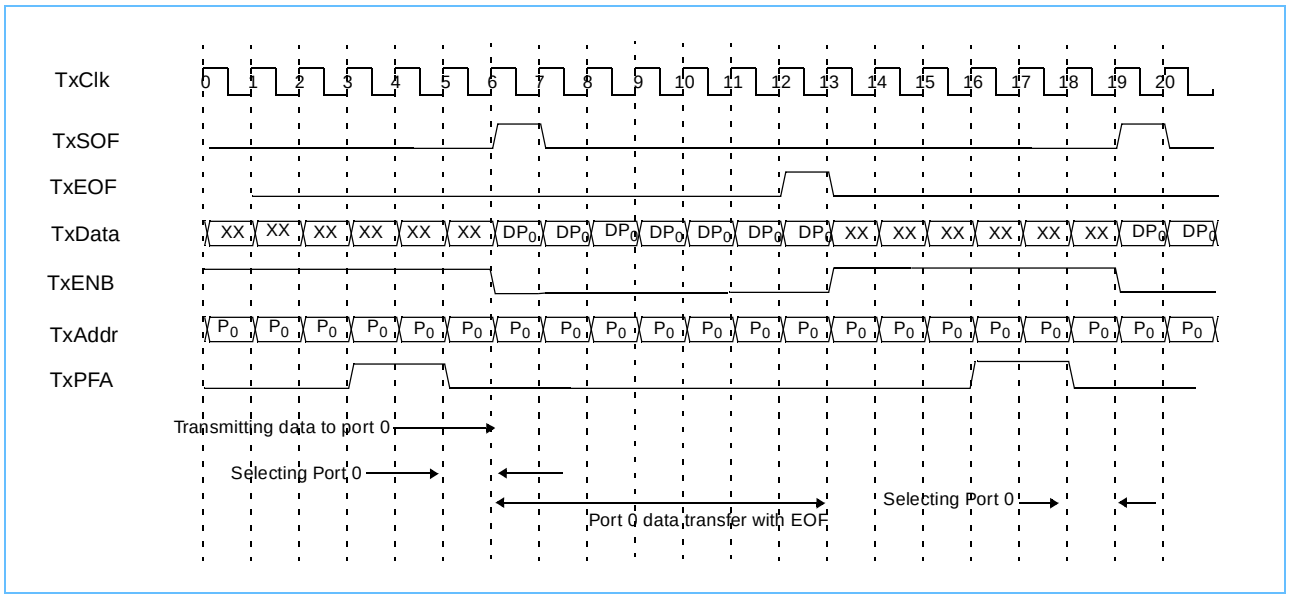


Figure 31: Transmit POS32 Interface Timing for 32-bit data bus (OC-48c).

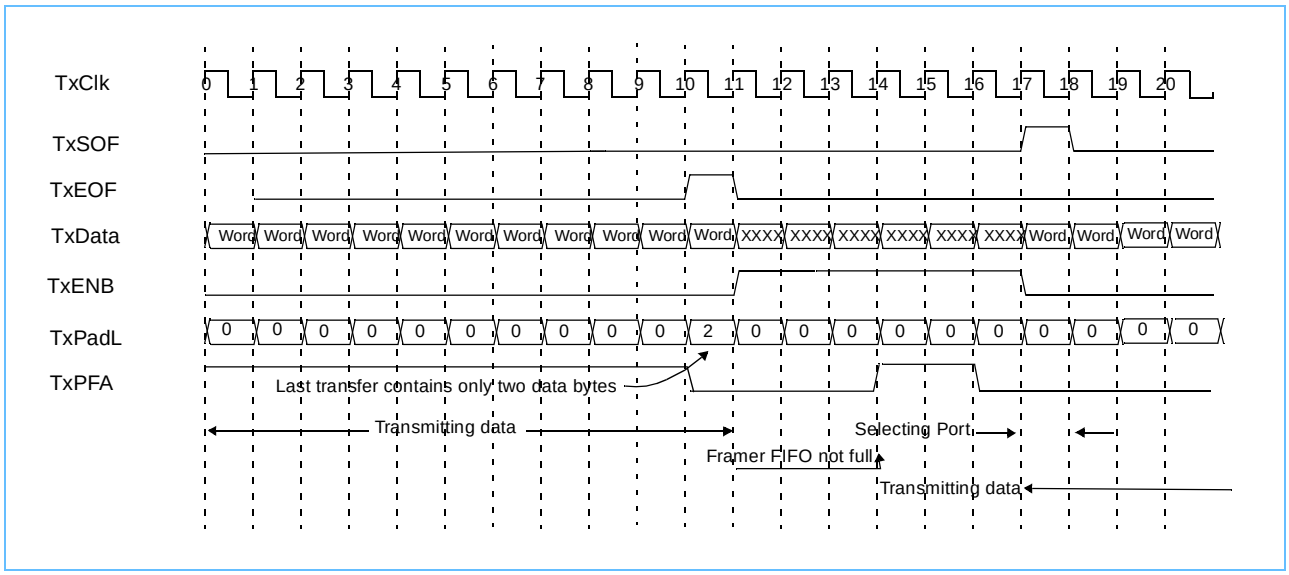


Figure 32: OC-3c/12/12c Configuration

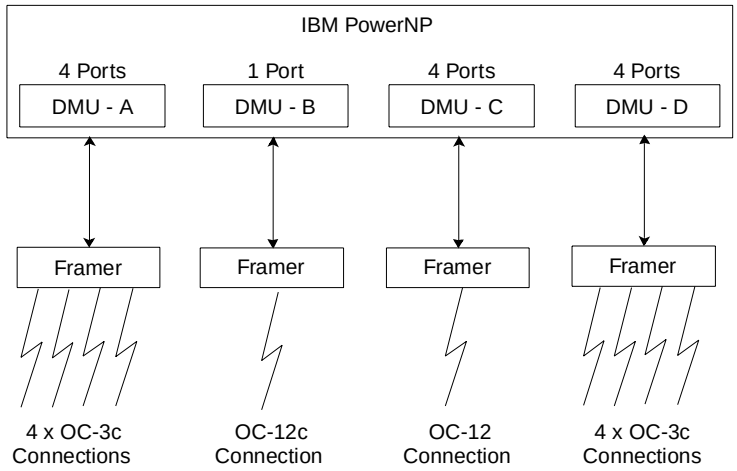


Figure 33 is an example of an OC-48 configuration in which each DMU is configured to operate in a single port mode to support OC-48. Each DMU supports an 8-bit data interface in both the ingress and egress directions.

Figure 33: OC-48 Configuration

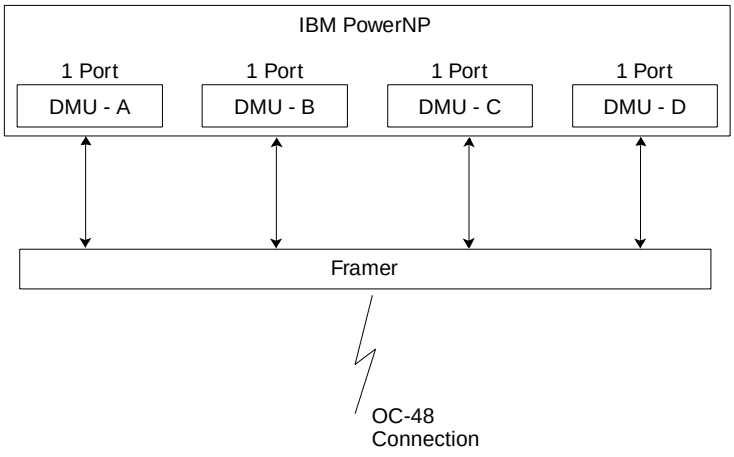


Figure 34 is an example of an OC-48c configuration in which DMU-A is configured to operate in a single port 32-bit mode to support OC-48c. Except for the I/Os, DMUs B, C, and D are not used in this configuration, although they must be configured for OC-48 mode and their data ports routed to DMU-A. The attached framer must also be configured for a 32-bit data interface.

Figure 34: OC-48c Configuration

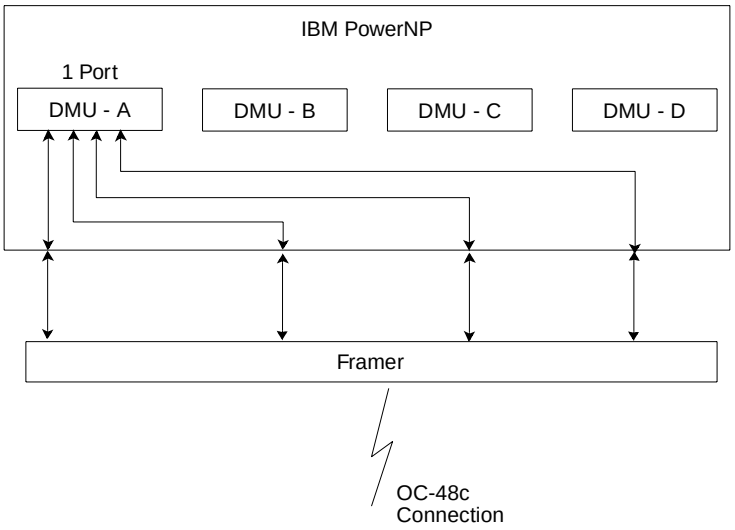


Table 41: POS Support

Feature	POS Mode	
	8-Bit	32-Bit
Supports quad port OC-3c connections, quad port OC-12 connections, or single port OC-12c connections	X	
Supports a single port OC-48c connection		X
Compatible with two vendor proprietary 8-bit POS implementations; Also compatible with 16-bit POS implementation using external logic	X	
Compatible with one vendor proprietary 32-bit POS implementations		X
Programmable CRC Insertion on a frame basis. With CRC Insertion disabled, the MAC transmits frames as is (suitable for switch environments) With CRC Insertion enabled, the MAC calculates and inserts the CRC	X	X
The minimum frame length the NP4GS3 can handle at a sustainable rate is 18 bytes. Smaller frames can be handled, but will consume the bandwidth of an 18-byte frame. An indirect back-pressure between the processor and the framer prevents frames from being lost, unless many small frames are received back-to-back.	X	X
Provides the following nine Tx Counters for testing and debugging purposes: number of bytes transmitted / received number of frames transmitted / received number of long frames transmitted / received number of frames with bad CRC transmitted / received number of receive errors	X	X

4. Ingress Enqueuer / Dequeuer / Scheduler

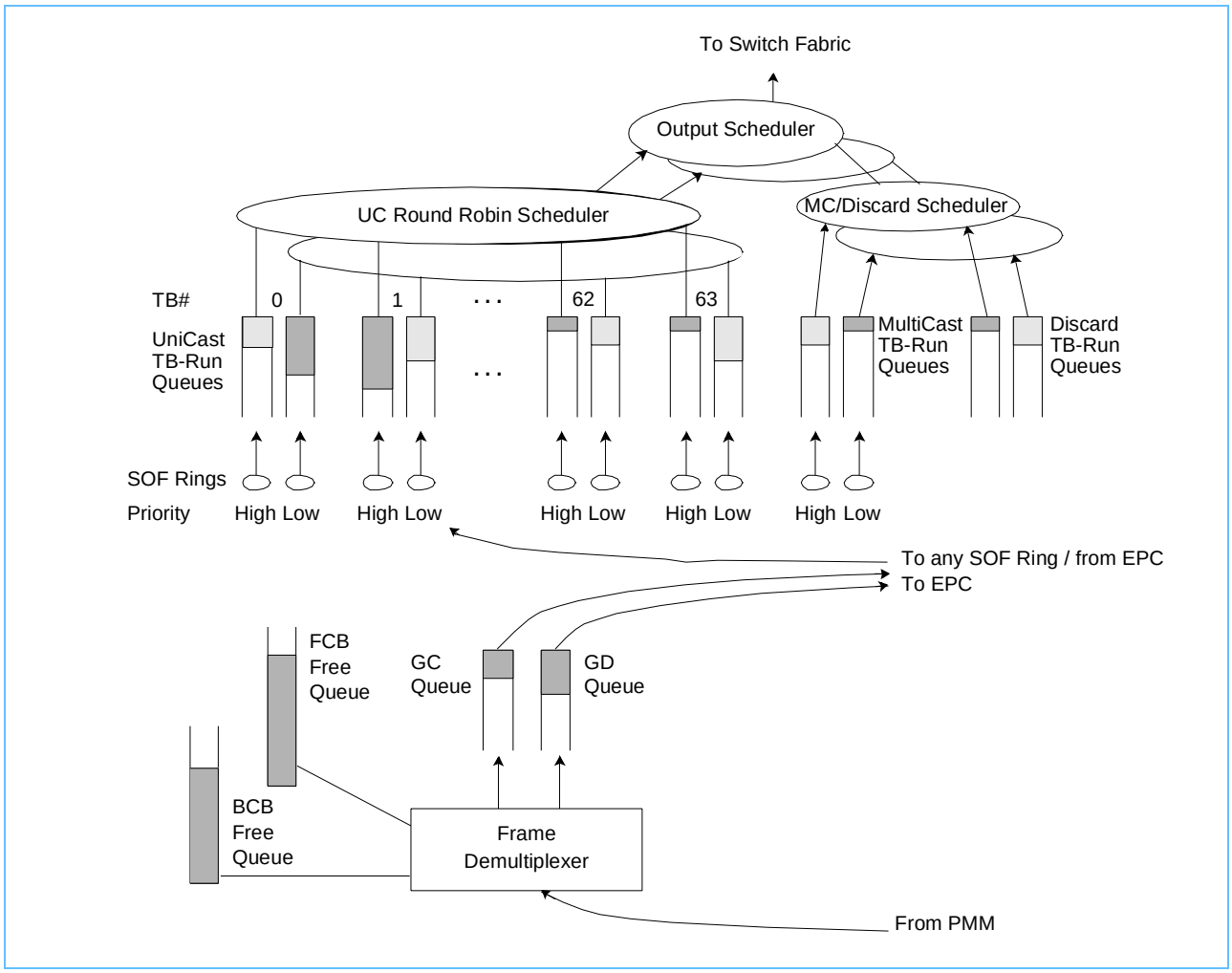
4.1 Overview

The Ingress Enqueuer / Dequeuer / Scheduler (Ingress-EDS) has interfaces with the PMM, the EPC, and the Ingress Switch interface. Frames that have been received by the media are passed through the PMM to the Ingress EDS. The Ingress EDS collects the frame data in its internal data store; when sufficient data has been received the frame is enqueued for processing to the EPC.

Once the frame is processed by the EPC, forwarding and QoS information is provided to the Ingress EDS. Hardware configured flow control mechanisms are invoked and the frame is then either discarded or placed into a queue to wait for transmission.

The Ingress EDS schedules all frames that cross the switch interface. When a frame has been selected, the frame data is passed to the Ingress Switch interface where the frame is segmented into data cells and sent on the DASL interface.

Figure 35: Logical Organization of the Data Flow Managed by the Ingress EDS



4.2 Ingress Flow Control

4.2.1 Overview

Flow control (whether to forward or discard frames) in the network processor is provided by hardware assist mechanisms and by picocode that implements a selected flow control algorithm. In general, flow control algorithms require information about the congestion state of the data flow, the rate at which packets arrive, the current status of the data store, the current status of target blades, and so on. A transmit probability for various flows is an output of these algorithms.

There are two implementations of flow control in the network processor:

- Flow control that is invoked when the frame is enqueued to a SOF queue. The hardware assist mechanisms use the transmit probability along with tail drop congestion indicators to determine if a forwarding or discard action should be taken during frame enqueue operation. The flow control hardware uses the picocode's entries in the ingress transmit probability memory to determine what flow control actions are required.
- Flow control that is invoked when frame data enters the network processor. When the ingress data store is sufficiently congested, these flow control actions discard either all frames, all data frames, or new data frames. The thresholds that control the invocation of these actions are BCB_FQ Threshold for Guided Traffic, BCB_FQ_Threshold_0, and BCB_FQ_Threshold_1 (see *Table 42: List of Flow Control Hardware Facilities* on page 113 for more information).

4.2.2 Flow Control Hardware Facilities

The hardware facilities listed in *Table 42* are provided for the picocode's use when implementing a flow control algorithm. The picocode uses the information from these facilities to create entries in the ingress transmit probability memory. The flow control hardware uses these entries when determining what flow control actions are required.

Table 42: List of Flow Control Hardware Facilities

Name	Definition	Access
BCB Free Queue (BCB_FQ) Control Block Register	Provides an instantaneous count of the number of free buffers available in the Ingress Data Store.	CAB
BCB_FQ Threshold for Guided Traffic	Threshold for BCB_FQ. When BCB_FQ < BCB_FQ_GT_Th, no further buffers are allocated for incoming data.	CAB
BCB_FQ_Threshold_0	Threshold for BCB_FQ. When BCB_FQ < BCB_FQ_Threshold_0, then no further buffers are allocated for incoming user traffic. Guided traffic can still allocate new buffers. When this threshold is violated, an interrupt (Class 0, bit 0) is signaled.	CAB
BCB_FQ_Threshold_1	Threshold for BCB_FQ. When BCB_FQ < BCB_FQ_Threshold_1, then no further buffers are allocated for new incoming user traffic. Guided traffic and packets already started can still allocate new buffers. When this threshold is violated, an interrupt (Class 0, bit 1) is signaled.	CAB
BCB_FQ_Threshold_2	BCB_FQ < BCB_FQ_Threshold_2, an interrupt (Class 0, bit 2) is signaled.	CAB
Flow Control Ingress Free Queue Threshold (FQ_P0_Th)	Threshold for BCB_FQ used when determining flow control actions against Priority 0 traffic. When BCB_FQ < FQ_P0_Th, the flow control hardware discards the frame.	CAB
Flow Control Ingress Free Queue Threshold (FQ_P1_Th)	Threshold for BCB_FQ used when determining flow control actions against Priority 1 traffic. When BCB_FQ < FQ_P1_Th, the flow control hardware discards the frame.	CAB
BCB FQ Arrival Count	Arrival rate of data into the ingress data store. This counter increments each time there is a dequeue from the BCB free queue. When read by picocode, via the CAB, this counter is set to 0 (Read with Reset).	CAB
Ingress Free Queue Count Exponentially Weighted Moving Average	Calculated EWMA of the BCB FQ (BCB_FQ_EWMA).	CAB
Flow Control Ingress Free Queue Threshold (FQ_SBFQ_Th)	Threshold for BCB_FQ. When BCB_FQ < FQ_SBFQ_TH, then the I_FreeQ_Th is set to 1. This is may be used by external devices assisting in flow control.	CAB
LocalTB_MC_Status_0	Threshold status of the priority 0 target DMU queues and the multicast queue. The status of the DMU_QCB.QCnt > DMU_QCB.Th tests are combined into a single result per target blade and the corresponding bit in LocalTB_MC_Status_0 is set. When the QCnt exceeds the threshold (Th), the bit is set to 1; otherwise it is set to 0.	Hardware only
Target DMU Queue Control Block (DMU_QCB)	Threshold for number of frames enqueued to the target DMU. These values are used to compare against the queue count when setting LocalTB_MC_Status_0 bits.	CAB
Remote TB Status 0	This 64-bit register contains the congestion status of all remote target blade's egress data stores. The congestion status of each remote target blade is the result of a comparison between the configured threshold and the EWMA of the offered rate of priority 0 traffic (See *Table 50: List of Flow Control Hardware Facilities* on page 136). This information is collected via the res_data IO.	Hardware only
Remote TB Status 1	This 64-bit register contains the congestion status of all remote target blade's egress data stores. The congestion status of each remote target blade is the result of a comparison between the configured threshold and the EWMA of the offered rate of priority 1 traffic (See *Table 50: List of Flow Control Hardware Facilities* on page 136). This information is collected via the res_data IO.	Hardware only

4.2.3 Hardware Function

4.2.3.1 Exponentially Weighted Moving Average (EWMA)

The hardware generates EWMA values for the BCB_FQ count, thus removing the burden of this calculation from the picocode. In general, EWMA for a counter X is calculated as follows:

```
EWMA_X = (1-K) * EWMA_X + K * X
```

This calculation occurs for a configured sample period and K ∈ {1, 1/2, 1/4, 1/8}.

4.2.3.2 Flow Control Hardware Actions

When the picocode enqueues a packet to be transmitted to a target blade, the flow control hardware examines the state of the FQ_P0_Th and FQ_P1_Th threshold status and the priority of the enqueued packet to determine what flow control action is required.

- If the FCInfo field of the FCBPage of the enqueued packet is set to x'F', flow control is disabled and the packet is forwarded.
- For priority 0 packets, if FQ_P0_TH or LocalTB_MC_Status_0 or Remote TB Status 0 is exceeded, then the packet will be discarded (tail drop discard). A counter block must be set up by the picocode to count these discards.
- For priority 1 packets, if FQ_P1_TH is exceeded, then the packet will be discarded (tail drop discard). Otherwise the transmit probability table is accessed and the value obtained is compared against a random number (∈ { 0 .. 1}) generated by the hardware. When the transmit probability is less than the random number, the packet is discarded.

  The index into the transmit probability table is QQQTCC where:

QQQ	QoS Class taken from the DSCP (Ingress FCBpage TOS field bits 7:5)
T	Remote TB Status 1 bit corresponding to the target blade. (Zero when the frame is multicast.)
CC	DSCP assigned color (Ingress FCBpage FCInfo field bits 1:0)

5. Switch Interface

5.1 Overview

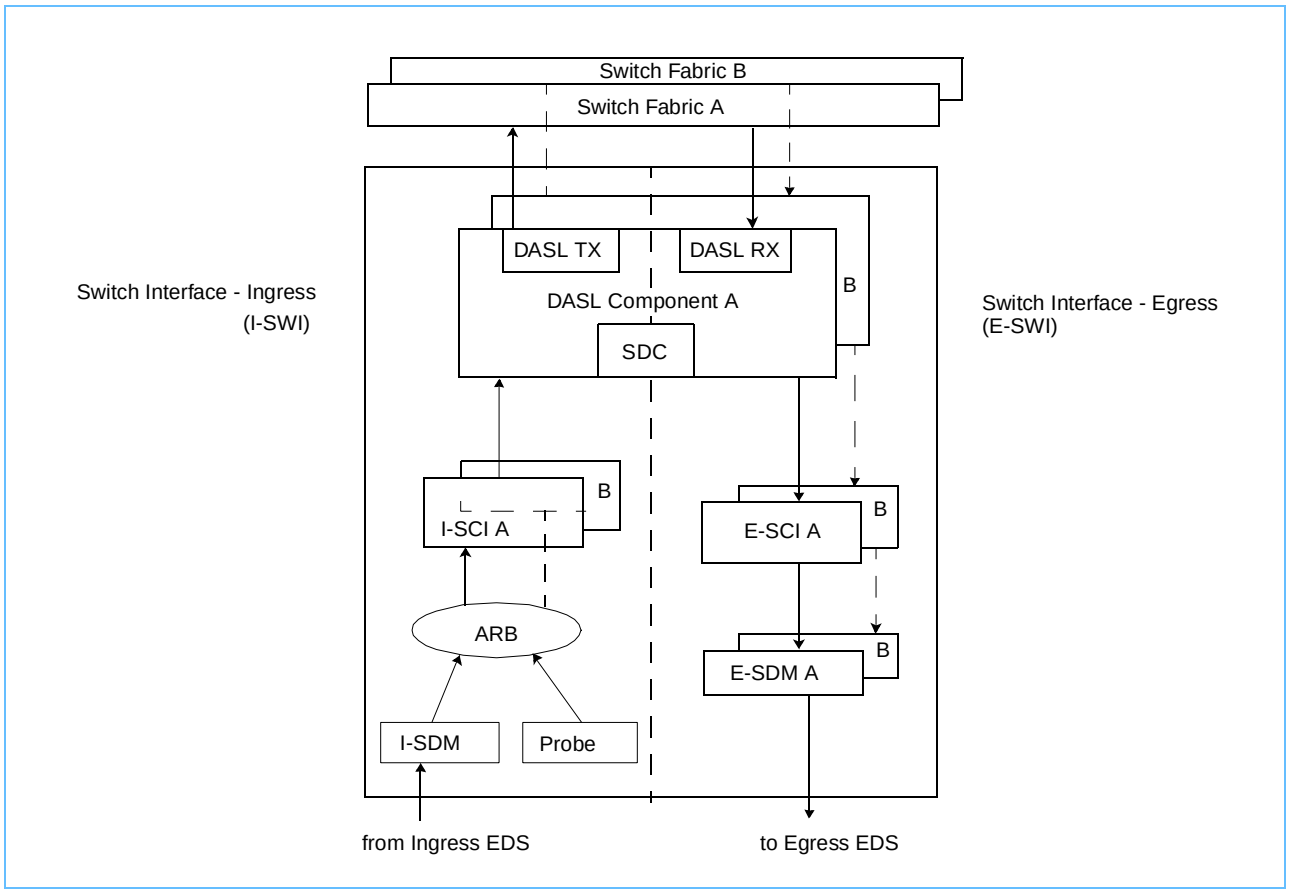
The Switch Interface (SWI) supports two high-speed DASL interfaces to attach to other network processors or to an external switch fabric. Each DASL link supports up to 3.25 to 4 Gbps throughput. The DASL links can be used in parallel (for example, to interconnect two network processors as dual network processors) or with one as the primary switch interface and the other as an alternate switch interface (for increased system availability). The DASL interfaces enable up to 64 network processors to be interconnected using an external switch fabric.

The SWI supports the following:

- Two DASL switch interfaces
- Simultaneous transfer of cells on both switch interfaces in both ingress and egress directions
- Building a Cell Header and Frame Header for each frame
- Segmenting a frame into 64-byte switch cells
- Cell Packing

The SWI's ingress side sends data to the switch fabric and its egress side receives data from the switch fabric. The DASL bus consists of four paralleled links: one Master and three Slaves. The network processor supports two DASL channels (A and B). An arbiter sits between the Ingress Switch Data Mover (I-SDM) and Ingress Switch Cell Interfaces SCI_A-1 and SCI_B-1. The arbiter directs the data traffic from the I-SDM and Probe to the appropriate I-SCI, based on switch_BnA chip input and the settings in the DASL Configuration Register (see *13.28.1 DASL Configuration Register (DASL_Config)* on page 400). *Figure 36* on page 116 shows the main functional units of the SWI.

Figure 36: Switch Interface Functional Units



The main units, described in following sections, are:

- I-SDM: Ingress Switch Data Mover
- I-SCI: Ingress Switch Cell Interface
- DASL: Data-Aligned Synchronous Links
- E-SCI: Egress Switch Cell Interface
- E-SDM: Egress Switch Data Mover

5.2 Ingress Switch Data Mover (I-SDM)

The I-SDM is the logical interface between the Ingress Enqueuer/Dequeuer/Scheduler's (EDS) frame data flow and the switch fabric's cell data flow. The I-SDM segments the frames into cells and passes the cells to the I-SCI.

The Ingress EDS controls the I-SDM data input by requesting the I-SDM to transmit data to the switch. To do this, the Ingress EDS gives the I-SDM the control information necessary to transmit one segment (either 48 or 58 bytes) of data from the selected frame. The control information consists of a Buffer Control Block (BCB) Address, Frame Control Block (FCB) record contents, and the BCB Address of the next frame going to the same switch destination (used for packing). With this information, the I-SDM builds a switch cell under a quadword-aligned format compatible with the data structure in the Egress Data Store. The logical switch cell structure contains three fields: Cell Header, Frame Header, and Data.

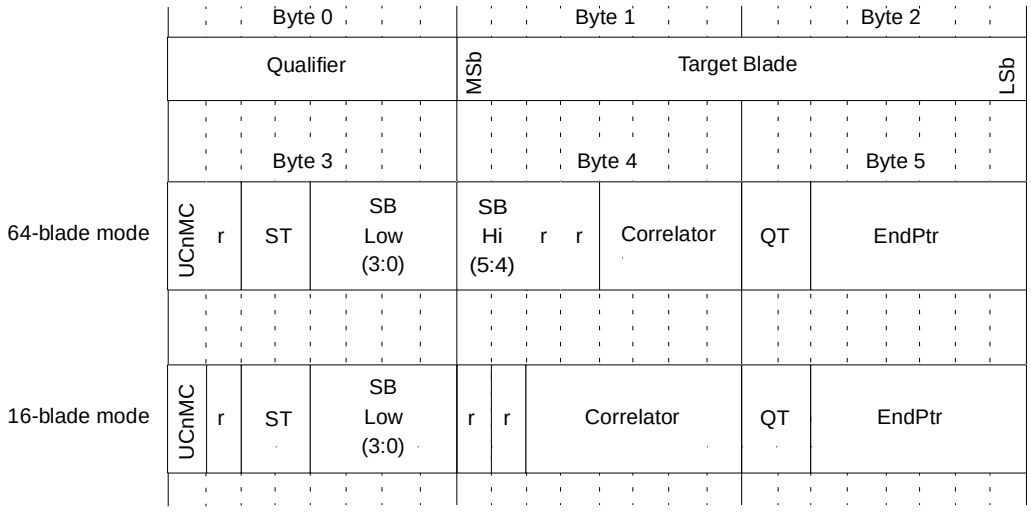
The first cell of a frame contains a cell header, a frame header, and 48 bytes of frame data. Following cells of a frame contain a cell header followed by 58 bytes of frame data. The final cell of a frame contains the cell header, remaining bytes of frame data, and any necessary bytes to pad out the remainder of the cell.

To increase the effective bandwidth to the switch, the network processor implements frame packing where the remaining bytes of a final cell contain the frame header and beginning bytes of the next frame. Packing is used when the target blade and priority of the next frame are the same as the target blade and priority of the preceding frame. In this case, the packed cell contains a 6-byte cell header, the remaining data bytes of the first frame, a 10-byte frame header of the second frame, and some data bytes of the second frame. Packing of frames occurs on 16-byte boundaries within a cell.

5.2.1 Cell Header

This is a 6-byte field holding control information for the cell. The first 3 bytes are used by the switch fabric for routing and flow control. The last 3 bytes are used by the target network processor.

Figure 37: Cell Header Format



The cell header is sent with each cell across the attached switch fabric. The fields of the cell header illustrated in *Figure 37* are described in *Table 43*.

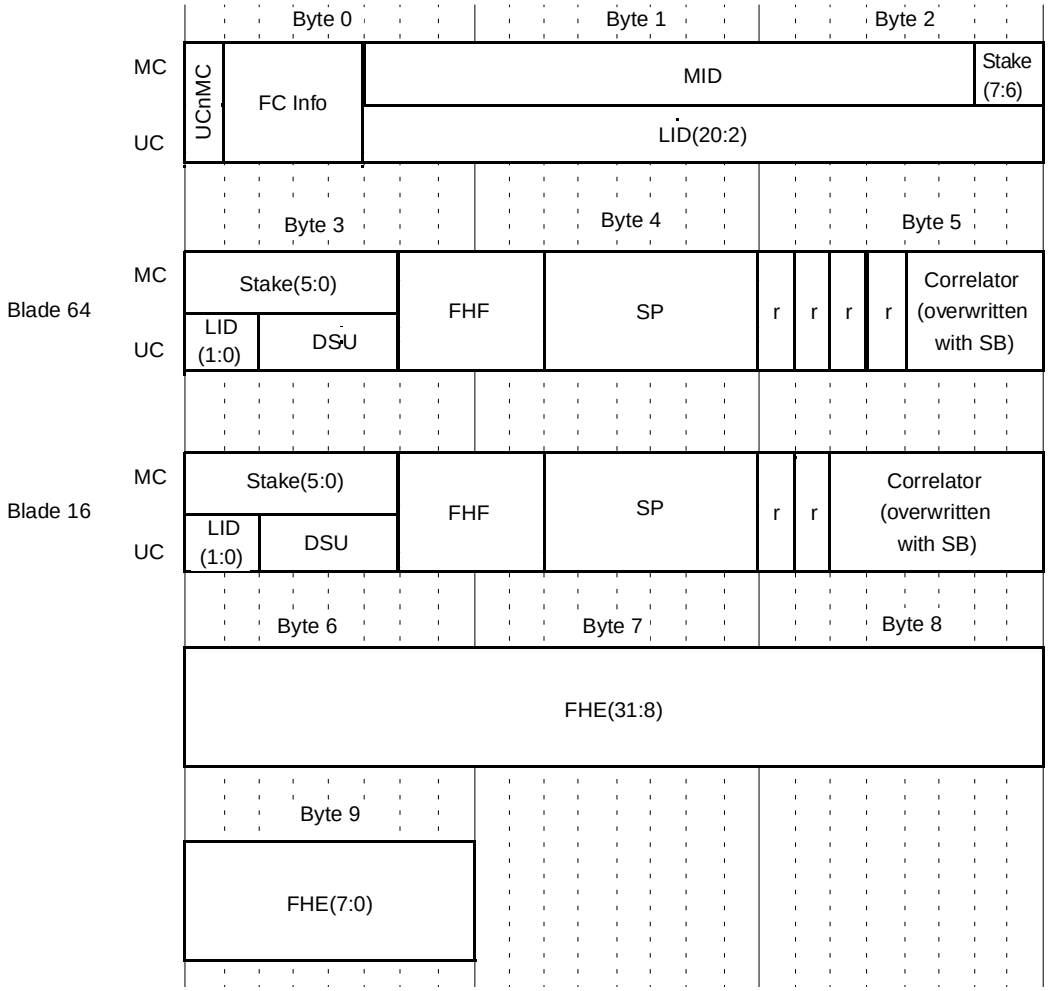
Table 43: Cell Header Fields

Field Name	Definition
Qualifier	This 8-bit field indicates the type of cell that is being transmitted. 7,2 Cell format value. The type of cell format used for this cell. '10' Frame format '11' Packed frame format All other values are reserved. 6 Switch cell header parity. This even parity bit covers the Qualifier and the TB fields. When a parity error is detected, the cell is discarded and the event is counted. 5:4 Data cell indicator. '00' Idle cell '11' Data cell All other values are reserved. 3 Reserved. Set to '0'. 1:0 Cell Priority '00' Highest priority - used for port mirroring traffic '01' High priority traffic '10' Low priority traffic '11' Reserved
Target Blade	Target blade address (16-bit field). Encoding of this field depends on the target blade mode. 16-blade The Target Blade field is a bit map of the destination target blades. Target Blade 0 corresponds to the MSb of the Target Blade field. 64-blade Valid unicast target blade field encodes are 0 through 63. Multicast encodes are in the range of 512 through 65535.
ST	Frame State Indicator (2-bit field). Provides information about the status of the frame currently being carried in the cell. 00 Continuation of current frame 01 End of current frame 10 Start of new frame 11 Start and end of new frame. This code point is also used to indicate a reassembly abort command. Abort is indicated when the End Pointer field value is '0'.
Source Blade (SB)	Source blade address. The size of this field depends on the blade operational mode and indicates the value of the Source blade.
Correlator	The size of this field depends on the blade operational mode. 16-blade Correlator values 0-63 are used for unicast traffic. Correlator values 0-31 are used for multicast traffic. 64-blade Correlator values 0-15 are used for unicast traffic. Correlator values 0-7 are used for multicast traffic.
QT(1:0)	Queue Type (2-bit field). Used to determine the required handling of the cell and the frame. Bits 1:0 Description 0x User traffic that is enqueued into data queue. 1x Guided traffic that is enqueued into the guided traffic queue. x0 Cell may be dropped due to switch congestion. x1 Cell may not be dropped due to switch congestion. QT(1) is set by the hardware when FC Info field of the frame header is set to x'F'.
EndPtr	End Pointer (6-bit field). Location of the last data byte in the cell when the State field indicates end of current frame (State = '01' or '11'). When an abort command is indicated (State must be '11'), the End Pointer is set to '0'. In all other cases (State = '00' or '10'), this field contains sequence checking information used by the frame reassembly logic in the network processor.
r	Reserved field, transmitted as '0' . Should not be modified or checked by the switch fabric.

5.2.2 Frame Header

This is a 10-byte field containing control information used by the target network processor and is sent once per frame.

Figure 38: Frame Header Format



The fields of the frame header illustrated in *Figure 38* are described in *Table 44*.

Table 44: Frame Header Fields

Field Name	Description
UC	Unicast. This 1-bit field indicates the format of the frame header. '0' Multicast format. '1' Unicast format.
FC Info	Flow Control Information (4-bit field). Indicates the type of connection used for this frame. Connection type encoding is used by the network processor's flow control mechanisms.
LID	Lookup Identifier. Used by the egress processing to locate the necessary information to forward the frame to the appropriate target port with the correct QoS.

Table 44: Frame Header Fields

Field Name	Description
MID	Multicast Identifier. Used by the egress processing to locate the multicast tree information which is used to forward the frame to the appropriate target ports with the correct QoS.
Stake	Available only for the multicast format of the frame header. This 8-bit field is used by egress processing to locate the start of the Layer 3 header.
DSU	Available only for the unicast format of the frame header. This 4-bit field indicates which egress data stores are used by this frame. Defined as follows (where "r" indicates a reserved bit that is transmitted as '0' and is not modified or checked on receipt): 0rr0 Data store 0 0rr1 Data store 1 1rr0 Data store 0 and data store 1 1rr1 Data store 0 and data store 1
FHF	Frame Header Format. Software controlled field. This field, with the addition of the UC field, is used by the Hardware Classifier in the EPC to determine the code entry point for the frame. The UC field and the FHF form a 5-bit index into a configurable lookup table of code entry points used for egress processing.
SP	Source port of the frame.
FHE	Frame Header Extension (32-bit field). Used by ingress processing to pass information to egress processing. The contents of this field depend on the FHF value used by egress processing to interpret the field. Information passed reduces the amount of frame parsing required by egress processing when determining how to forward the frame.
Correlator	The size of this field depends on the blade operational mode. The SB information in the cell header is copied into the byte occupied by the correlator (overlaying the correlator and some reserved bits) when the frame header is written to the egress data store. This provides SB information for egress processing. 16-blade Correlator values 0-63 are used for unicast traffic. Correlator values 0-31 are used for multicast traffic. 64-blade Correlator values 0-15 are used for unicast traffic. Correlator values 0-7 are used for multicast traffic.

5.3 Ingress Switch Cell Interface (I-SCI)

The I-SCI provides a continuous stream of transmit data to the DASL. After power on or reset of the network processor, the I-SCI initialization process generates synchronization cells for the DASL. When the DASL is synchronized, the I-SCI enters the operational state. When there are no data cells to send, the I-SCI generates idle cells for transmission via the DASL. When there are data cells to send, the I-SCI receives a logical cell from the I-SDM, translates the cell into the switch cell format, and transmits the cell via the DASL. The I-SCI performs formatting on the cell headers to conform with the switch cell format.

The I-SCI unit relies on an internal FIFO of cells, written by the I-SDM at the network processor core clock rate, and read at the Switch clock rate performing the translation from logical cell format to switch cell format.

5.3.1 Idle Cell Format

When there is no data to send, idle cells are transmitted on the switch interface. All bytes in an idle cell have the value x‘CC’, except for the first 3 bytes in the master stream. Word 0, Word 1, and Word 2 of the master stream contain H0, H1, and H2 respectively.

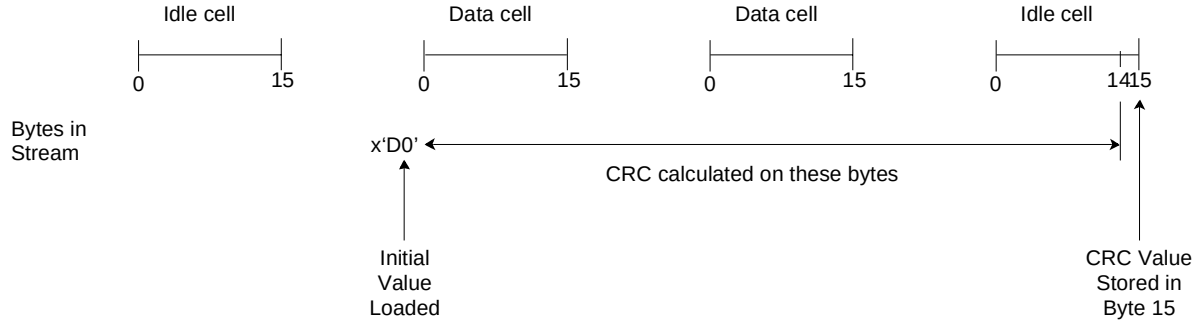
Table 45: Idle Cell Format Transmitted to the Switch Interface

Word #	Slave 1 (Byte 0)	Master (Byte 1)	Slave 3 (Byte 2)	Slave 2 (Byte 3)
	(bits 31:24)	(bits 23:16)	(bits 15:8)	(bits 7:0)
W0	x‘CC’	H0	x‘CC’	x‘CC’
W1	x‘CC’	H1 (x‘CC’)	x‘CC’	x‘CC’
W2	x‘CC’	H2 (x‘CC’)	x‘CC’	x‘CC’
W3	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W4	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W5	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W6	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W7	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W8	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W9	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W10	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W11	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W12	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W13	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W14	x‘CC’	x‘CC’	x‘CC’	x‘CC’
W15	CRC	CRC	CRC	CRC

5.3.1.1 CRC Bytes: Word 15

The CRC bytes sent on each byte stream contain an 8-bit CRC checksum of the polynomial $X^8+X^4+X^3+X^2+1$. Each byte stream is independently calculated and the initial value loaded into the CRC generator at the end of each Idle cell (after Word 15) is x‘D0’. The final CRC value is calculated starting with the first byte following an Idle Cell up to (but not including) the byte sent as part of Word 15 in the next Idle Cell.

Figure 39: CRC Calculation Example



5.3.1.2 I-SCI Transmit Header for an Idle Cell

The idle cell header consists of 3 bytes, H0-H2, and are defined as follows:

- H0: Also referred to as the Qualifier byte, this 8-bit field indicates the type of cell being transmitted.
- H1 and H2: set to x'CC'.

Bit(s)	Definition
7	Reserved. Transmitted as '0'.
6	Switch cell header parity. This even parity bit covers the H0-H2 fields.
5:4	Data cell indicator. '00' Idle cell '11' Data cell All other values are Reserved.
3:2	Reserved. Transmitted as '00'.
1:0	Reserved. This field should not be examined by the switch.

5.4 Switch Data Cell Format - Ingress and Egress

The table below shows the format of data cells sent to and from the Switch interface via the Data-Aligned Synchronous Link bus. Notice that the Packet Routing Switch cell header bytes (H0, H1, and H2) are all contained in the master byte stream. Bytes designated CH0 through CH5 are the cell header bytes described in *Figure 37: Cell Header Format* on page 117.

Table 46: Switch Data Cell Format

Word #	Slave 1 DASL_Out_0 DASL_Out_1	Master DASL_Out_2 DASL_Out_3	Slave 3 DASL_Out_4 DASL_Out_5	Slave 2 DASL_Out_6 DASL_Out_7
	(bits 31:24)	(bits 23:16)	(bits 15:8)	(bits 7:0)
W0	CH5	CH0 (H0)	D9	D7
W1	D0	CH1 (H1)	D4	D2
W2	D1	CH2 (H2)	D5	D3
W3	CH4	CH3	D8	D6
W4	D25	D23	D21	D19
W5	D12	D10	D16	D14
W6	D13	D11	D17	D15
W7	D24	D22	D20	D18
W8	D41	D39	D37	D35
W9	D28	D26	D32	D30
W10	D29	D27	D33	D31
W11	D40	D38	D36	D34
W12	D57	D55	D53	D51
W13	D44	D42	D48	D46
W14	D45	D43	D49	D47
W15	D56	D54	D52	D50

1. DASL_Out_x (where x ∈ {1,3,5,7}) carry the even bits of the indicated bytes
2. DASL_Out_x (where x ∈ {0, 2, 4, 6}) carry the odd bits of the indicated bytes
3. Dxx indicates the data byte in the cell from the SDM interface
4. CHx indicates the cell header

5.5 Data-Aligned Synchronous Link (DASL)

The Data-Aligned Synchronous Link (DASL) interface is a macro that facilitates high speed point-to-point inter-chip communication. It provides the application designer the ability to relieve I/O constraints imposed by the chip package or card connector. The DASL interface performs multi-bit serialization and de-serialization to reduce the I/O pin count. The interface is frequency synchronous which removes the need for asynchronous interfaces that introduce additional interface latency. DASL links are intended to operate over a backplane without any additional components.

The DASL interface macro was developed to interface the core area of a CMOS ASIC to a high speed link. The macro incorporates all the high-speed circuitry necessary to initialize and perform dynamic link timing and data serialization/de-serialization without exposing the core designer to the specific implementation. The core ASIC interface to DASL is a parallel interface. The DASL macro is scalable based on application needs.

The DASL macro was designed for high levels of integration. It uses reduced voltage differential transceivers to reduce power consumption. The macro has been partitioned such that multiple sub-macros share a common controller as well as a common phase locked loop (PLL).

On the most basic level, the DASL macro is designed to provide a 4 to 1 serialization/de-serialization interface. The NP4GS3 application utilizes a 32- to 8-bit (32-bit port) pin reduction. The shared DASL controller (SDC) provides control for the DASL TX and DASL RX ports.

5.6 Egress Switch Cell Interface (E-SCI)

The E-SCI receives cells from the switch fabric and passes data cells to the E-SDM. The E-SCI discards idle cells received from the switch fabric, checks the parity of the received cells, and discards cells with bad parity. The E-SCI strips out the target blade per-port grant information from the switch cells and sends this grant information to the network processor Ingress units for use in ingress data flow control. The E-SCI assists in egress data flow control by throttling the switch fabric to prevent data overruns.

5.6.1 Output Queue Grant (OQG) Reporting

Most fields in the cell header must pass through the switch fabric unchanged (idle cells are not transmitted through a switch fabric; they originate at the output port of the switch). The exception to this is the Target Blade field (see *Figure 37*) which contains Output Queue Grant (OQG) information used by the network processor's Ingress EDS when selecting data to be sent to the switch. A target port is not selected if its corresponding Output Queue Grant information is set to ‘0’.

Further, Output Queue Grant information is sent for each of the three priorities supported by the network processor. This is done using multiple cell transfers; starting with priority 0, reporting Output Queue Grant for all supported target blades, then repeating this process for each priority level until a value of 2 is reached, and starting over at priority 0 again.

For 16-blade operational mode the sequence is:

1. Priority 0; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
2. Priority 1; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
3. Priority 2; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
4. Start over at step 1.

The idle cell indicates the priority of the OQG it carries. This allows the network processor to synchronize with the external switch.

For 64-blade operational mode the sequence is:

1. Priority 0; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
2. Priority 0; OQG for blades 16:31 (H1 contains 16:23, H2 contains 24:31)
3. Priority 0; OQG for blades 32:47 (H1 contains 32:39, H2 contains 40:47)
4. Priority 0; OQG for blades 48:63 (H1 contains 48:55, H2 contains 56:63)
5. Priority 1; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
6. Priority 1; OQG for blades 16:31 (H1 contains 16:23, H2 contains 24:31)
7. Priority 1; OQG for blades 32:47 (H1 contains 32:39, H2 contains 40:47)
8. Priority 1; OQG for blades 48:63 (H1 contains 48:55, H2 contains 56:63)
9. Priority 2; OQG for blades 0:15 (H1 contains 0:7, H2 contains 8:15)
10. Priority 2; OQG for blades 16:31 (H1 contains 16:23, H2 contains 24:31)
11. Priority 2; OQG for blades 32:47 (H1 contains 32:39, H2 contains 40:47)
12. Priority 2; OQG for blades 48:63 (H1 contains 48:55, H2 contains 56:63)
13. Start over at step 1.

For 64-blade mode, this sequence is interrupted when an idle cell is sent. *Table 49* illustrates an idle cell that carries the OQG for all blades. Status for each priority is given in increasing order; (P0 followed by P1 followed by P2). The network processor starts the above sequence at step 1 with the arrival of the next data cell.

5.6.2 Switch Fabric to Network Processor Egress Idle Cell

Egress Switch Interface requires idle cells when there is no data. An idle cell requires:

- The last bytes on each stream contain a Trailer CRC.
- For the master stream, the header H0 as described in *Table 47* below.
- For the master stream, H1- H2 contain the target blade grant priority information when in 16-blade mode.
- For the master stream, H1-H2 contain x‘CC’ when in 64-blade mode.
- All other bytes contain x’CC’ when in 16-blade mode or OQG (see *Table 48* on page 126) when in 64-blade mode.

Table 47: Receive Cell Header Byte H0 for an Idle Cell

Bit(s)	Definition
7	Reserved. Transmitted as '0'.
6	Switch cell header parity. This even parity bit covers the H0-H2 fields.
5:4	Data cell indicator. '00' Idle cell '11' Data cell All other values are Reserved.
3:2	Reserved. Transmitted as '00'.
1:0	Output Queue Grant priority. When in 16-blade mode, indicates the priority level of the Output Queue Grant information carried in H1-H2.Otherwise the network processor ignores this field.

Table 48: Idle Cell Format Received From the Switch Interface - 16-blade Mode

Slave 1	Master 1	Slave 3	Slave 2
Byte 0	Byte 1	Byte 2	Byte 3
x'CC'	H0	x'CC'	x'CC'
x'CC'	H1 (OQG (0:7))	x'CC'	x'CC'
x'CC'	H2 (OQG (8:15))	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
x'CC'	x'CC'	x'CC'	x'CC'
CRC	CRC	CRC	CRC

Table 49: Idle Cell Format Received From the Switch Interface - 64-blade Mode

Slave 1								Master								Slave 3								Slave 2							
Byte 0								Byte 1								Byte 2								Byte 3							
x‘CC’								H0								x‘CC’								x‘CC’							
x‘CC’								H1 (x‘CC’)								x‘CC’								x‘CC’							
x‘CC’								H2 (x‘CC’)								x‘CC’								x‘CC’							
0	1	$\overline{0}$	$\overline{1}$	2	3	$\overline{2}$	$\overline{3}$	4	5	$\overline{4}$	$\overline{5}$	6	7	$\overline{6}$	$\overline{7}$	8	9	$\overline{8}$	$\overline{9}$	10	11	$\overline{10}$	$\overline{11}$	12	13	$\overline{12}$	$\overline{13}$	14	15	$\overline{14}$	$\overline{15}$
16	17	$\overline{16}$	$\overline{17}$	18	19	$\overline{18}$	$\overline{19}$	20	21	$\overline{20}$	$\overline{21}$	22	23	$\overline{22}$	$\overline{23}$	24	25	$\overline{24}$	$\overline{25}$	26	27	$\overline{26}$	$\overline{27}$	28	29	$\overline{28}$	$\overline{29}$	30	31	$\overline{30}$	$\overline{31}$
32	33	$\overline{32}$	$\overline{33}$	34	35	$\overline{34}$	$\overline{35}$	36	37	$\overline{36}$	$\overline{37}$	38	39	$\overline{38}$	$\overline{39}$	40	41	$\overline{40}$	$\overline{41}$	42	43	$\overline{42}$	$\overline{43}$	44	45	$\overline{44}$	$\overline{45}$	46	47	$\overline{46}$	$\overline{47}$
48	49	$\overline{48}$	$\overline{49}$	50	51	$\overline{50}$	$\overline{51}$	52	53	$\overline{52}$	$\overline{53}$	54	55	$\overline{54}$	$\overline{55}$	56	57	$\overline{56}$	$\overline{57}$	58	59	$\overline{58}$	$\overline{59}$	60	61	$\overline{60}$	$\overline{61}$	62	63	$\overline{62}$	$\overline{63}$
0	1	$\overline{0}$	$\overline{1}$	2	3	$\overline{2}$	$\overline{3}$	4	5	$\overline{4}$	$\overline{5}$	6	7	$\overline{6}$	$\overline{7}$	8	9	$\overline{8}$	$\overline{9}$	10	11	$\overline{10}$	$\overline{11}$	12	13	$\overline{12}$	$\overline{13}$	14	15	$\overline{14}$	$\overline{15}$
16	17	$\overline{16}$	$\overline{17}$	18	19	$\overline{18}$	$\overline{19}$	20	21	$\overline{20}$	$\overline{21}$	22	23	$\overline{22}$	$\overline{23}$	24	25	$\overline{24}$	$\overline{25}$	26	27	$\overline{26}$	$\overline{27}$	28	29	$\overline{28}$	$\overline{29}$	30	31	$\overline{30}$	$\overline{31}$
32	33	$\overline{32}$	$\overline{33}$	34	35	$\overline{34}$	$\overline{35}$	36	37	$\overline{36}$	$\overline{37}$	38	39	$\overline{38}$	$\overline{39}$	40	41	$\overline{40}$	$\overline{41}$	42	43	$\overline{42}$	$\overline{43}$	44	45	$\overline{44}$	$\overline{45}$	46	47	$\overline{46}$	$\overline{47}$
48	49	$\overline{48}$	$\overline{49}$	50	51	$\overline{50}$	$\overline{51}$	52	53	$\overline{52}$	$\overline{53}$	54	55	$\overline{54}$	$\overline{55}$	56	57	$\overline{56}$	$\overline{57}$	58	59	$\overline{58}$	$\overline{59}$	60	61	$\overline{60}$	$\overline{61}$	62	63	$\overline{62}$	$\overline{63}$
0	1	$\overline{0}$	$\overline{1}$	2	3	$\overline{2}$	$\overline{3}$	4	5	$\overline{4}$	$\overline{5}$	6	7	$\overline{6}$	$\overline{7}$	8	9	$\overline{8}$	$\overline{9}$	10	11	$\overline{10}$	$\overline{11}$	12	13	$\overline{12}$	$\overline{13}$	14	15	$\overline{14}$	$\overline{15}$
16	17	$\overline{16}$	$\overline{17}$	18	19	$\overline{18}$	$\overline{19}$	20	21	$\overline{20}$	$\overline{21}$	22	23	$\overline{22}$	$\overline{23}$	24	25	$\overline{24}$	$\overline{25}$	26	27	$\overline{26}$	$\overline{27}$	28	29	$\overline{28}$	$\overline{29}$	30	31	$\overline{30}$	$\overline{31}$
32	33	$\overline{32}$	$\overline{33}$	34	35	$\overline{34}$	$\overline{35}$	36	37	$\overline{36}$	$\overline{37}$	38	39	$\overline{38}$	$\overline{39}$	40	41	$\overline{40}$	$\overline{41}$	42	43	$\overline{42}$	$\overline{43}$	44	45	$\overline{44}$	$\overline{45}$	46	47	$\overline{46}$	$\overline{47}$
48	49	$\overline{48}$	$\overline{49}$	50	51	$\overline{50}$	$\overline{51}$	52	53	$\overline{52}$	$\overline{53}$	54	55	$\overline{54}$	$\overline{55}$	56	57	$\overline{56}$	$\overline{57}$	58	59	$\overline{58}$	$\overline{59}$	60	61	$\overline{60}$	$\overline{61}$	62	63	$\overline{62}$	$\overline{63}$
CRC								CRC								CRC								CRC							

5.6.3 Receive Header Formats for Sync Cells

Sync cells are a special type of idle cells. They are similar to the received idle cell for 16-blade mode (see *Table 48* on page 126) but H0, H1, and H2 are set to a value of x‘CC’. All sync cells are discarded.

5.7 Egress Switch Data Mover (E-SDM)

The E-SDM is the logical interface between the switch cell data flow of the E-SCI and the frame data flow of the Egress EDS. The E-SDM serves as a buffer for cells flowing from the E-SCI to the Egress EDS. The E-SDM extracts control information, such as the frame correlator, which is passed to the Egress EDS. The Egress EDS uses this control information, along with data from the E-SDM, to re-assemble the cells into frames.

6. Egress Enqueuer / Dequeuer / Scheduler

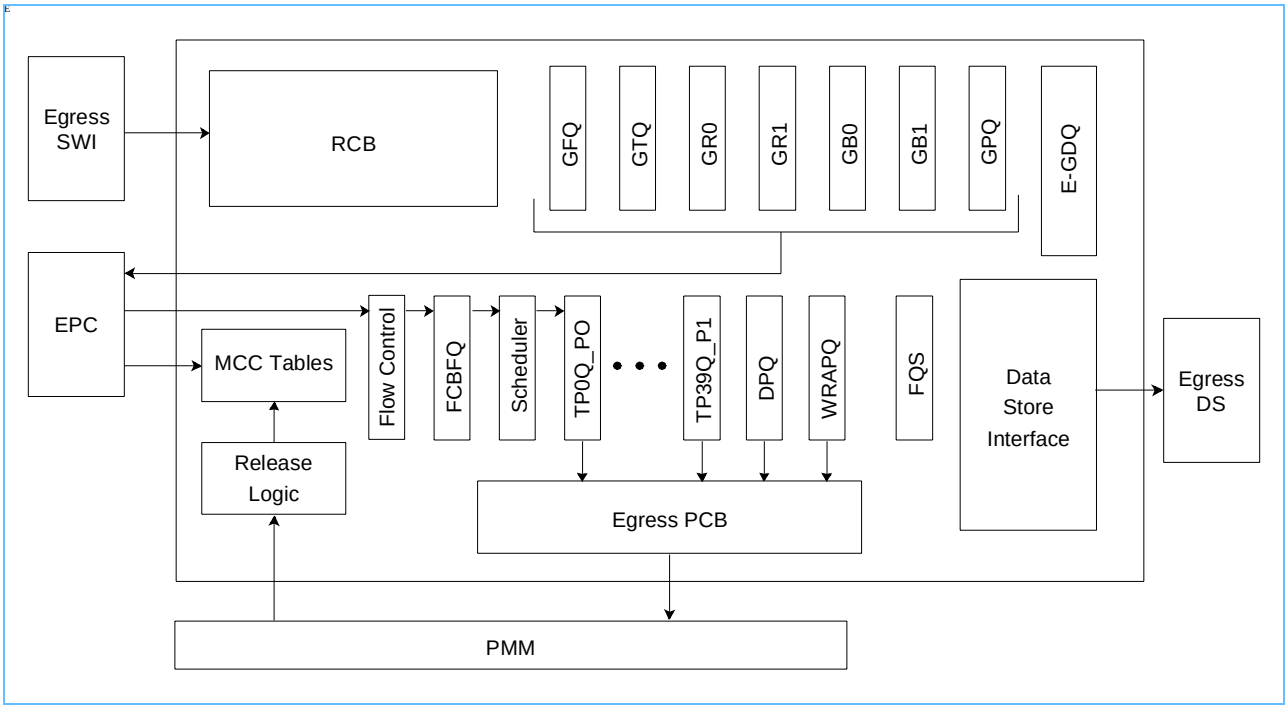
6.1 Overview

The Egress Enqueuer / Dequeuer / Scheduler (Egress EDS) supports reassembly of frames sent from up to 64 network processors up to a total of 3072 simultaneous reassemblies. Each switch cell received from the Egress Switch Interface is examined and stored in the appropriate Egress Data Store (Egress DS) for reassembly into its original frame. When the frame is completely received from the switch interface, the Egress EDS enqueues the frame to the EPC for processing. The EPC processes the frame and then enqueues it to either the scheduler, when enabled, or to a target port (TP) queue for transmission out the Egress PMM. The Egress EDS is responsible for handling all the buffer, frame, and queue management for reassembly and transmission on the egress side of the network processor.

The Egress EDS supports the following:

- External Egress Data Store
- Automatic allocation of buffer and frame control blocks for each frame
- EPC queues (GFQ, GTQ, GR0, GR1, GB0, GB1, and GPQ)
- Target Port queues with two priorities
- Buffer Thresholds and flow control actions
- Bandwidth and best effort scheduling
- Reading and writing of frame data stored in Egress Data Store
- Up to 512 K Buffer twins depending on memory configuration
- Up to 512 K of FCBs depending on memory configuration
- Unicast and multicast frames
- Cell Packing
- 40 external ports plus wrap port interface to the PMM
- Discard function
- Hardware initialization of internal and external data structures

Figure 40: Egress EDS Block Diagram See Egress EDS Components for term definitions



6.1.1 Egress EDS Components

Data Store Interface	Writes the external Egress Data Stores during frame reassembly and reads them during frame transmission. Also gives the EPC access to the Egress Data Stores. The Data Store Interface supports two external data stores: DS0 and DS1.
DPQ	Discard Port Queue. Releases twin buffers back to the free queue stack. This is used by the picocode to discard frames where header twins have been allocated.
E-GDQ	Discard Queue Stack. Holds frames that need to be discarded. This is used by the hardware to discard frames when the egress DS is congested or to re-walk a frame marked for discard for a half duplex port. Used by the picocode to discard frames that do not have header twins allocated.
Egress PCB	Egress Port Control Block. Contains all the information needed to send a frame to the Egress PMM for transmission. The Egress EDS uses this information to walk the twin buffer chain and send the data to the Egress PMM's output port. There is a PCB entry for each target port, plus one for Discard and one for Wrap. Each entry holds information for two frames: the current frame being sent to the PMM port and the next frame to be sent.
FCBFQ	Frame Control Block Free Queue. Lists free Egress FCBs. FCBs store all the information needed to describe the frame on the Egress side, such as starting buffer, length, MCC address, frame alteration, and other enqueue information. The Egress EDS obtains an FCB from the free queue when the EPC enqueues the frame to either a flow QCB (see *Flow Queues* on page 144) or a target port after EPC processing and flow control actions are complete.

FQS	Free Queue Stack. Holds a list of free Egress twin buffers which are used by the Egress EDS during frame reassembly and by the EPC during frame alteration. The twin buffers store the frame data or frame alteration header twins and also contain the link pointer to the next buffer in the chain. The FQS is used to obtain a new free twin buffer any time the Egress needs one and to return free twin buffers after frames are discarded or transmitted.
GB0, GB1	Low Priority Data Queues. GB0 is for frames stored in Egress DS0. GB1 is for frames stored in Egress DS1.
GFQ	Guided Frame Queue. Queue that contains guided frames for delivery for the Egress side of the network processor to the Guided Frame Handler.
GPQ	PowerPC queue. Queue that contains frames re-enqueued for delivery to the GPH for processing.
GR0, GR1	High Priority Data Queues. GR0 is for frames stored in Egress DS0. GR1 is for frames stored in Egress DS1.
GTQ	General Table Queue. Queue that contains guided frames re-enqueued by pico-code for delivery to the GTH.
MCC Table	Multicast Count Table. Each entry stores the number of multicast frames associated with a particular set of twin buffers. If a frame is to be multicast to more than one target port, the EPC enqueues the frame to each target port causing an entry in the MCC Table to be incremented. As each target port finishes its copy of the frame, the MCC Table entry is decremented. When all ports have sent their copies of the frame, the associated twin buffers are released.
RCB	Reassembly Control Block. Used by the Egress EDS to reassemble the cells received from the switch fabric into their original frames. Contains pointers to the Egress DS to specify where the contents of the current cell should be stored. Helps the Egress EDS to keep track of the frame length and which EPC queue to use.
Release Logic	Releases twin buffers after the PMM has finished with the contents of the buffer. The Release Logic checks the MCC Table to determine if the buffer can be released or is still needed for some other copy of a multicast frame.
TP0Q - TP39Q	Target Port Queues. Hold linked lists of frames destined for a target port. Two queues are associated with each of the 40 possible target ports. These queues are prioritized from high (P0) to low (P1) using a strict priority service scheme (all higher priority queues within a target port set must be empty before starting a lower priority queue).
WRAPQ	Wrap Queue. Two wrap queues, one for guided frames and one for data frames, send frames from the Egress side to the Ingress side of the network processor. These queues allow the network processor to respond to a guided frame sent from a remote Control Point Function. The guided frame is received from the switch fabric on the Egress side and is wrapped to the Ingress side to allow a response to be sent to the CPF across the switch fabric. A data frame may be wrapped when processing on a remote CPF is required.

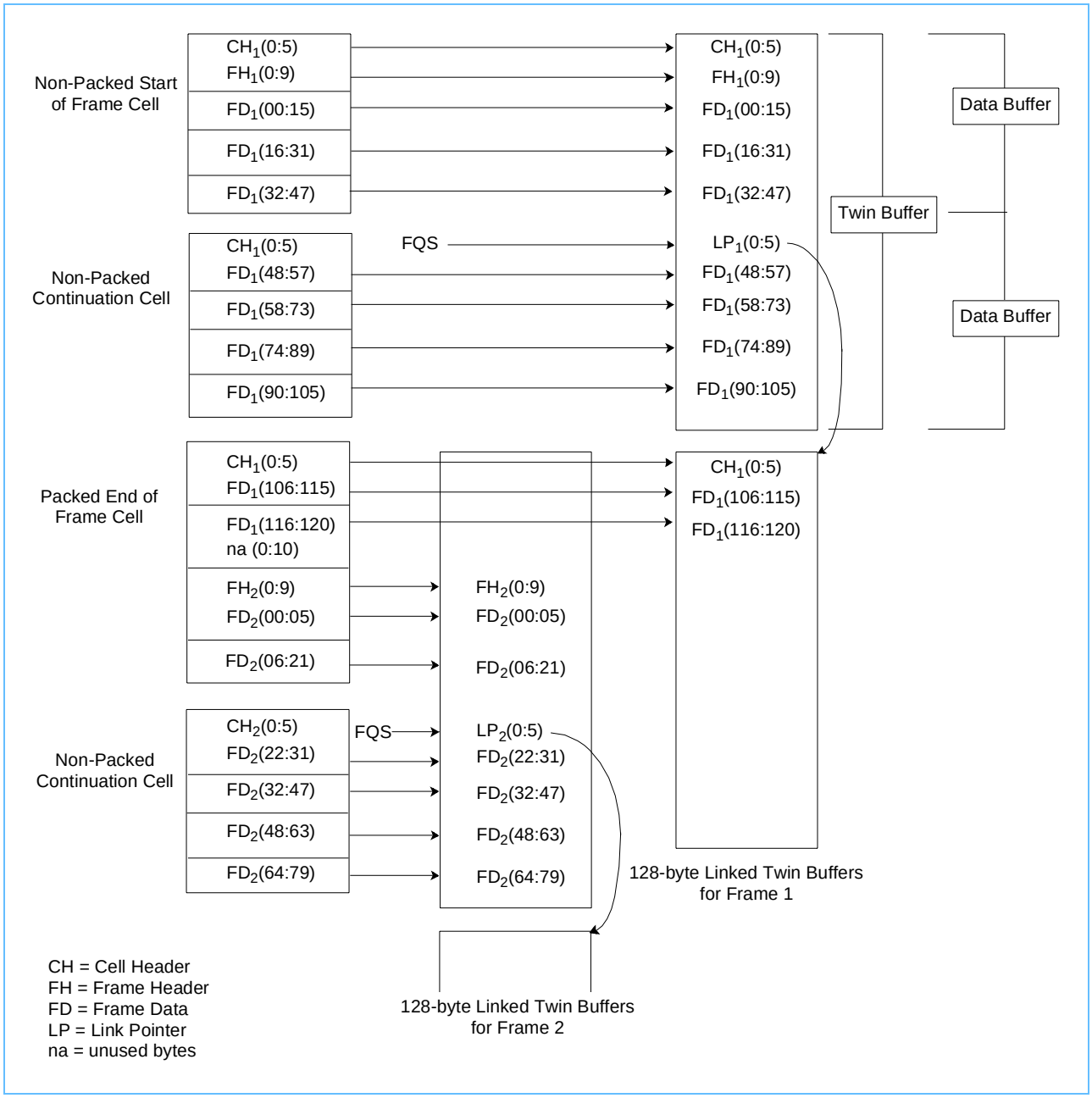
6.2 Operation

Switch cells are received from the Egress SWI along with information that has been preprocessed by the Egress SDM such as the Reassembly Correlator, Source Blade, multicast indication, priority, and target Data Store. In order to optimize the data transfer to and Egress DS, the Egress EDS uses the target Data Store's information to determine which Egress SDM to service. The two Egress DSs, DS0 and DS1, use alternating write windows, meaning the Egress EDS can write one cell to one data store each "cell window time" (the time needed to store an entire switch cell (64 bytes) in external DRAM). Cell window time is configured using the DRAM Parameter Register's 11/10 field.

After the appropriate Egress SDM has been selected, the Egress EDS reads the cell data out of the SDM and uses the Reassembly Correlator, source blade, multicast indication, and priority information to index into the RCB. The RCB contains all the information needed to reassemble the frame, including the buffer address to use in the Egress DS. The cell data is stored in the buffer and the RCB is updated to prepare for the next cell associated with this same frame. The Egress EDS manages 3072 RCB entries and each entry contains information such as start-of-frame indicator, data store buffer address, current reassembled frame length, and queue type. Cells from several source blades are interleaved coming from the switch interface and the Egress EDS uses the RCB information to rebuild each frame.

The Egress EDS uses a free buffer from the head of the FQS as needed to store frame data in the Egress DS. The Egress EDS stores the cell data in the appropriate buffer and also stores the buffer chaining information over the cell header data. When a packed cell arrives, the Egress EDS stores each frame's information in two separate twin buffers. The first portion of the packed cell contains the end of a frame. This data is stored in the appropriate twin buffer as pointed to by the RCB. The remaining portion of the packed cell will be the beginning of another frame and this data will be stored in a second twin buffer as indicated by the second frame's RCB entry.

Figure 41: Cell Formats and Storage in Egress Data Store

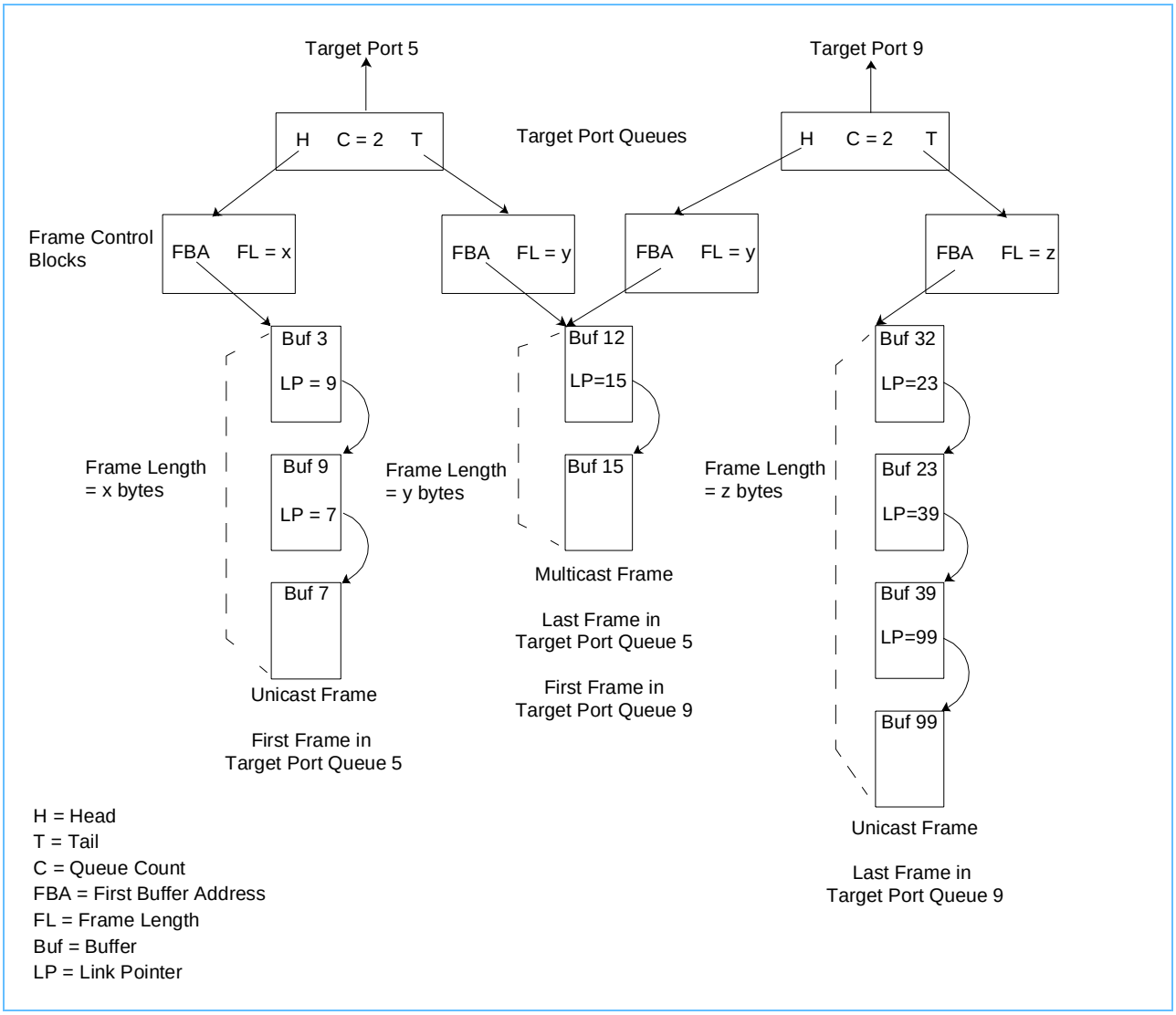


When the entire frame is reassembled, the Egress EDS enqueues the frame to one of several EPC queues. If the reassembled frame is a guided frame, the Egress EDS uses the GFQ. High priority frames are placed in either the GR0 or the GR1. Low priority frames are placed in either the GB0 or the GB1. The EPC services these queues and requests a programmable amount of read data from the various frames in order to process them (see *Table 126: Port Configuration Memory Index* on page 200). The Egress EDS reads the data from the Egress Data Store and passes it back to the EPC. Additional reads or writes can occur while the EPC is processing the frame. The Egress EDS performs all necessary reads or writes to the Egress Data Store as requested by the EPC.

When the frame has been processed, the EPC enqueues the frame to the Egress EDS. If the frame's destination is the General Table Handler, the Egress EDS enqueues the frame to the GTQ. If the frame is to be discarded, it is placed in the DPQ. If the frame needs to be wrapped to the Ingress side, it is placed in the Wrap Queue. All other frames are subject to flow control actions. If flow control does not discard the frame, the frame is placed into the scheduler, if enabled, or placed directly into a target port queue. Each target port supports two priorities and therefore has two queues. The EPC indicates which target port and which priority should be used for each frame enqueued. The two queues per port use a strict priority scheme, which means that all high priority traffic must be transmitted before any lower priority traffic will be sent.

Frames destined for the scheduler, target ports, or wrap ports have a Frame Control Block (FCB) assigned by the Egress EDS. The FCB holds all the information needed to transmit the frame including starting buffer address, frame length, and frame alteration information. If the EPC needs to forward more than one copy of the frame to different target ports, an entry in the MCC Table is used to indicate the total number of copies to send. The EPC enqueues the frame to different target ports or different flow QCBs and each enqueue creates a new FCB with (possibly) its own unique frame alteration.

Figure 42: TPQ, FCB, and Egress Frame Example



When a frame reaches the head of a target port queue, it is placed in the Egress Port Control Block entry (PCB) for that port and the FCB for that frame is placed on the FCB Free Queue. The Egress EDS uses the PCB to manage the frames that are being sent to the PMM for transmission. The PCB stores the information needed to retrieve frame data, such as current buffer address and frame length, from the Egress DS. The PCB allows up to 40 frames to be retrieved simultaneously from the Egress DS. The PCB also supports the Wrap Port and the Discard Port Queue. As data is retrieved from the Egress DS and passed to the PMM, the PCB monitors the transfers and stores the buffer link pointers that enable the PCB to walk the buffer chain for the frame. When the entire buffer chain has been traversed, the PCB entry is updated with the next frame for that target port.

As the PMM uses the data from each buffer, it passes the buffer pointer back to the Release Logic in the Egress EDS. The Release Logic examines the MCC Table entry to determine if the buffer should be returned to the FQS. (Half-Duplex ports will not have their twin buffers released until the entire frame has been transmitted, in order to to support the recovery actions that are necessary when a collision occurs on the Ethernet media.) If the MCC entry indicates that no other copies of this frame are needed, the buffer pointer is stored in the FQS. However, if the MCC entry indicates that other copies of this frame are still being used, the Egress EDS decrements the MCC entry, but does no further action with this buffer pointer.

The DPQ contains frames that have been enqueued for discard by the EPC and by the Release Logic. The hardware uses the DPQ to discard the last copy of frames transmitted on half duplex ports. The DPQ is dequeued into the PCB's discard entry, where the frame data is read from the DS to obtain buffer chaining information necessary to locate all twin buffers of the frame and to release these twin buffers back to the free pool (FQS).

6.3 Egress Flow Control

6.3.1 Overview

Flow control (whether to forward or discard frames) in the network processor is provided by hardware assist mechanisms and picocode that implements a selected flow control algorithm. In general, flow control algorithms require information about the congestion state of the data flow, including the rate at which packets arrive, the current status of the data store, the current status of target blades, and so on. A transmit probability for various flows is an output of these algorithms.

There are two implementations of flow control in the network processor:

- Flow control that is invoked when the frame is enqueued to either a target port queue or a flow QCB. The hardware assist mechanisms use the transmit probability along with tail drop congestion indicators to determine if a forwarding or discard action should be taken during frame enqueue operation. The flow control hardware uses the picocode's entries in the Egress transmit probability memory to determine what flow control actions are required.
- Flow control that is invoked when frame data enters the network processor. When the Egress DS is sufficiently congested, these flow control actions discard all frames. The threshold that controls the invocation of these actions is FQ_ES_Threshold_0 (see *Table 50: List of Flow Control Hardware Facilities* on page 136 for more information).

6.3.2 Flow Control Hardware Facilities

The hardware facilities listed in *Table 50* are provided for the picocode's use when implementing a flow control algorithm. The picocode uses the information from these facilities to create entries in the Egress transmit probability memory. The flow control hardware uses these entries when determining what flow control actions are required.

Table 50: List of Flow Control Hardware Facilities

Name	Definition	Access
Free Queue Count	Instantaneous count of the number of free twins available in the Egress Data Store.	CAB
FQ_ES_Threshold_0	Threshold for Free Queue Count. When the Free Queue Count < FQ_ES_Threshold_0, no further twins are allocated for incoming data. User packets that have started reassembly are discarded when they receive data when this threshold is violated. Guided traffic is not discarded. The number of packets discarded is counted in the Reassembly Discard Counter. When this threshold is violated, an interrupt (Class 0, bit 4) is signaled.	CAB
FQ_ES_Threshold_1	Threshold for Free Queue Count. When the Free Queue Count < FQ_ES_Threshold_1, an interrupt (Class 0, bit 5) is signaled.	CAB
FQ_ES_Threshold_2	Threshold for Free Queue Count. When the Free Queue Count < FQ_ES_Threshold_2, an interrupt (Class 0, bit 6) is signaled, and if enabled by DMU configuration, the Ethernet MAC preamble is reduced to 6 bytes.	CAB
Arrival Rate Counter	This is the arrival rate of data into the Egress Data Store. This counter increments each time there is a dequeue from the Twin Free Queue. When read by picocode, via the CAB, this counter is set to 0 (Read with Reset).	CAB
FQ Count EWMA	This is the calculated EWMA of the Free Queue Count	CAB
P0 Twin Count	This counter is the number of Twins in priority 0 packets that have been enqueued to flow queues, but have not been dequeued from target port queues.	CAB

Table 50: List of Flow Control Hardware Facilities (Continued)

Name	Definition	Access
P1 Twin Count	This counter is the number of twins in priority 1 packets that have been enqueued to flow queues, but have not been dequeued from target port queues.	CAB
P0 Twin Count Threshold	Threshold for P0 Twin Count which is used when determining flow control actions against Priority 0 traffic. When P0 Twin Count < P0 Twin Count Threshold, the flow control hardware will discard the frame.	CAB
P1 Twin Count Threshold	Threshold for P1 Twin Count which is used when determining flow control actions against Priority 1 traffic. When P1 Twin Count < P1 Twin Count Threshold, the flow control hardware will discard the frame.	CAB
Egress P0 Twin Count EWMA	This is the calculated EWMA of the count of the number of twins allocated to P0 traffic during the sample period. Hardware maintains a count of the number of twins allocated to P0 traffic at enqueue.	CAB
Egress P1Twin Count EWMA	This is the calculated EWMA of the count of the number of twins allocated to P1 traffic during the sample period. Hardware maintains a count of the number of twins allocated to P1 traffic at enqueue.	CAB
Egress P0 Twin Count EWMA Threshold	The congestion status of the Egress Data Store in each target blade is the result of a comparison between this configured threshold and the EWMA of the offered rate of priority 0 traffic (P0 Twin Count EWMA < P0 Twin Count EWMA threshold). This information is transmitted to remote blades via the res_data I/O and is collected in the Remote TB Status 0 register in the ingress flow control hardware.	CAB
Egress P1 Twin Count EWMA Threshold	The congestion status of Egress Data Store in each target blade is the result of a comparison between this configured threshold and the EWMA of the offered rate of priority 0 traffic. (P1 Twin Count EWMA < P1 Twin Count EWMA threshold). This information is transmitted to remote blades via the res_data I/O and is collected in the Remote TB Status 1 register in the ingress flow control hardware.	CAB
Target Port PQ+FQ_Th	Target Port port queue plus egress scheduler (flow QCB) threshold. The target port queues maintain a count of the number of twins allocated to the target port. The count is incremented on enqueue (after flow control transmit action is taken) and decremented on dequeue from the target port. Thresholds for each priority can be configured for for all ports (0:39). When the number of twins assigned to a target port queue exceeds the threshold, its threshold exceed status is set to 1. The status is used to index into the transmit probability memory for packets going to the target port.	CAB
QCB Threshold	Flow queue threshold. When the number of twins assigned to this flow queue exceeds this threshold, the threshold exceed status is set to 1. The status is used to index into the transmit probability memory.	CAB

6.3.3 Remote Egress Status Bus

6.3.3.1 Overview

The Remote Egress Status (RES) Bus communicates the congestion state of the Egress Datastores of all NP4GS3s in a system to the Ingress Flow Control of every NP4GS3 in that system. The ingress portion of each NP4GS3 can then preemptively discard frames destined for a congested NP4GS3 without consuming additional bandwidth through the switch.

6.3.3.2 Bus Sequence and Timing

The RES Bus consists of two bidirectional signals:

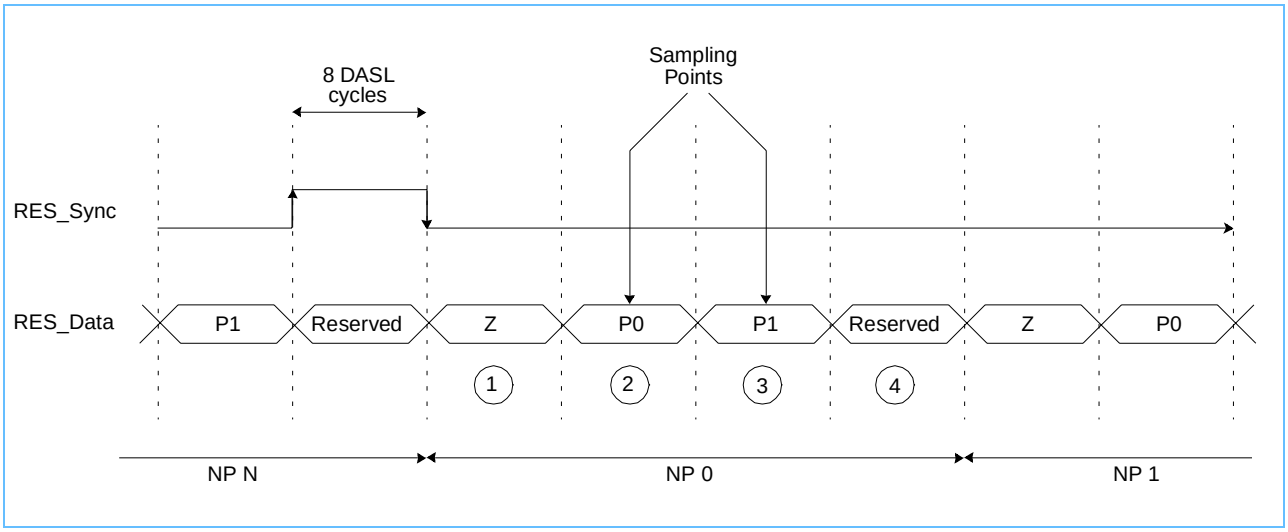
1. **RES_Data**. This signal is time-division multiplexed between all the NP4GS3s in a system. Only one NP4GS3 at a time drives this signal. Each NP4GS3 drives two priorities of congestion status sequentially. All of the NP4GS3s in a system sample this data and store it for use by the Ingress Flow Control to make

discard decisions.

2. **RES_Sync**. This signal is received by all NP4GS3s in a system. Each NP4GS3 samples the negative edge of this signal to derive its time slot to drive Egress Datastore congestion information. This signal is periodic and repeats to allow the NP4GS3s to resynchronize after 16 or 64 time slots have passed, depending on the value of the TB Mode register. In a system, one NP4GS3 can be configured to drive this signal to the other NP4GS3s. Alternatively, an external chip can provide the stimulus for this signal.

Figure 43 shows a timing diagram of the operation of the RES Bus.

Figure 43: RES Bus Timing



The RES Bus operates on the same clock frequency as the internal DASL clock. This clock period can range from 8 ns to 10 ns. The clock that the RES Bus uses is not necessarily in phase with the DASL clock. In addition, each NP4GS3 is not necessarily in phase with other NP4GS3s in the same system. The RES_Sync signal is responsible for synchronizing all the NP4GS3s.

The RES Bus is time-division multiplexed between every NP4GS3 in the system. Each NP4GS3 takes a turn driving its congestion information onto the RES Bus in the order of its My_TB register (i.e., the NP4GS3 with a My_TB setting of 0 drives immediately following the fall of RES_Sync, followed by the NP4GS3 with a My_TB setting of 1, etc.)

Within each NP4GS3 time slot, the NP4GS3 puts four values on the RES_Data signal. Each value is held for eight DASL clock cycles. Therefore, each NP4GS3 time slot is 32 DASL clock cycles. The protocol for sending the congestion information is as follows:

1. **High-Z** - The NP4GS3 keeps its RES_Data line in high impedance for eight DASL clock cycles. This allows the bus to turn around from one NP4GS3 to another.
2. **P0** - The NP4GS3 drives its Priority 0 (P0) Egress Datastore congestion information for eight DASL clock cycles. This status is high when the Egress P0 Twin Count EWMA register value is greater than the Egress P0 Twin Count EWMA Threshold register value. It is low otherwise.
3. **P1** - The NP4GS3 drives its Priority 1 (P1) Egress Datastore congestion information for eight DASL clock cycles. This status is high when the Egress P1 Twin Count EWMA register value is greater than the Egress P1 Twin Count EWMA Threshold register value. It is low otherwise.

4. **Reserved** - The NP4GS3 drives a low value for eight DASL clock cycles. This is reserved for future use.

The Ingress Flow Control samples RES_Data during the midpoint of its eight DASL clock cycles. This provides a large enough sample window to allow for jitter and phase differences between the DASL clocks of two different NP4GS3s.

The RES_Sync signal is driven high during the Reserved cycle of the last NP4GS3's time slot. The RES_Sync signal therefore has a period equal to 32 DASL clock periods multiplied by the number of NP4GS3s supported in this system. The number of NP4GS3s can be either 16 or 64 depending on the value of the TB Mode register.

6.3.3.3 Configuration

The RES Bus is activated by enabling the Ingress and Egress portions via the Remote Egress Status Bus Configuration Enables register. This register contains a bit to enable the Ingress logic to capture the RES_Data, a bit to enable the Egress logic to send its status on RES_Data, and a bit to enable one of the NP4GS3s to drive RES_Sync.

6.3.4 Hardware Function

6.3.4.1 Exponentially Weighted Moving Average (EWMA)

The hardware generates EWMA values for the Free queue count and the P0/P1 twin counts, thus removing the burden of this calculation from the picocode. In general, EWMA for a counter X is calculated as follows:

```
EWMA_X = (1-K) * EWMA_X + K * X
```

This calculation occurs for a configured sample period and K $\in$ {1, 1/2, 1/4, 1/8).

6.3.4.2 Flow Control Hardware Actions

When the picocode enqueues a packet to be transmitted to a target port, the flow control logic examines the state of the Target port PQ+FQ_Th, and the priority of the enqueued packet to determine if any flow control actions are required.

- If the FCInfo field of the FCBPage of the enqueued packet is set to x'F', flow control is disabled and the packet is forwarded without regard to any of the congestion indicators.
- For priority 0 packets, if Target port PQ+FQ_Th or P0 Twin Count Threshold is exceeded, then the packet will be discarded. A counter block must be set up by the picocode to count these discards.
- For priority 1 packets, if P1 Twin Count Threshold is exceeded, then the packet will be discarded. Otherwise the transmit probability found in the QCB (available only when the scheduler is enabled) and the transmit probability table are accessed. The smaller of these values is compared against a random number ($\in$ { 0 .. 1}) generated by the hardware. When the transmit probability is 0 or is less than the random number, the packet is discarded. A counter block must be set up by the picocode to count these discards.

The index into the transmit probability table is TTCCFP where:

TT	Packet type (Egress FCBpage FCInfo field bits 3:2).
CC	DSCP assigned color (Egress FCBpage FCInfo field bits 1:0)
F	Threshold exceeded status of the target flow queue (QCB threshold exceeded)
P	Threshold exceeded status of the target port queue (Target port Pq+FQ_th exceeded)

6.4 The Egress Scheduler

6.4.1 Overview

The Egress Scheduler provides shaping functions in the network processor. The Egress scheduler manages bandwidth on a per frame basis by determining the amount of bandwidth a frame requires (i.e. the number of bytes to be transmitted) and comparing this against the amount of bandwidth permitted by the configuration of the frame's flow queue. The amount of bandwidth used by a first frame affects when the scheduler permits the transmission of a second frame of a flow queue. The Egress Scheduler characterizes flow queues with the parameters listed in *Table 51*:

Table 51: Flow Queue Parameters

Parameter Name	Description
Low Latency Sustainable Bandwidth (LLS)	Provides guaranteed bandwidth with qualitative latency reduction.
Normal Latency Sustainable Bandwidth (NLS)	Provides guaranteed bandwidth. LLS has higher service priority over NLS, such that Flow Queues connected to LLS have better latency characteristics when compared to Flow Queues connected to NLS.
Peak Bandwidth Service (PBS)	Provides additional bandwidth on a best effort basis.
Queue Weight	Allows the scheduler to assign available (i.e. not assigned or currently not used by LLS and NLS) bandwidth to flow queues using best effort or PBS services. Assignment of different ratios of the available bandwidth is accomplished by assigning different queue weights to queues that share the same target port.

A graphical representation of the Egress Scheduler is shown in *Figure 44: The Egress Scheduler* on page 142. A list of valid combinations of above parameters is given in *Table 52: Valid Combinations of Scheduler Parameters* on page 143.

Figure 44: The Egress Scheduler

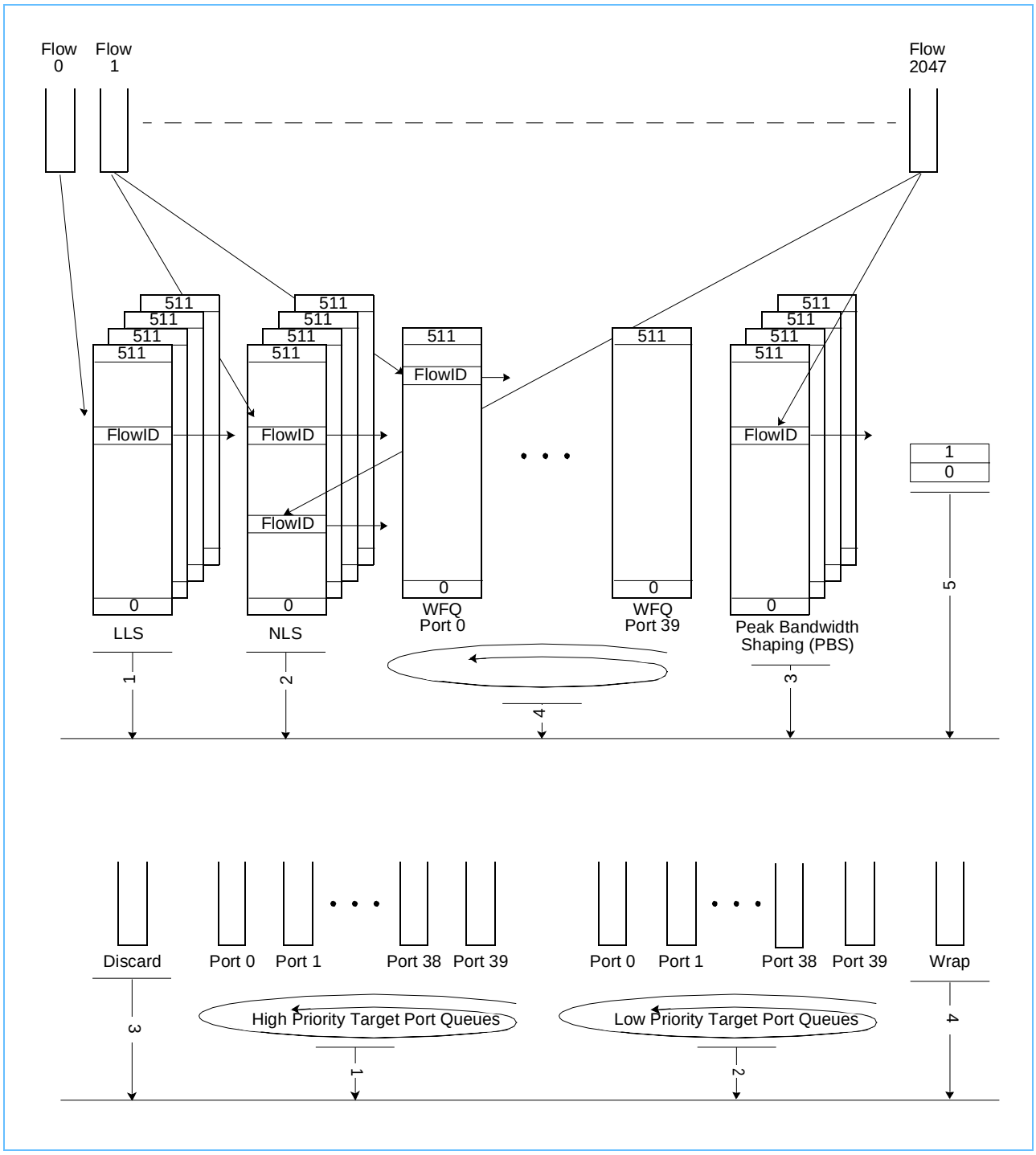


Table 52: Valid Combinations of Scheduler Parameters

QOS	LLS	NLS	Weight	PBS
Low Latency with Guaranteed BW Shaping	X			
Normal Latency with Guaranteed BW Shaping		X		
Best Effort			X	
Best Effort with Peak Rate			X	X
Normal Latency with Guaranteed BW Shaping and Best Effort		X	X	
Normal Latency with Guaranteed BW Shaping and Best Effort and Peak Rate		X	X	X

6.4.2 Egress Scheduler Components

The egress scheduler consists of the following components:

- Scheduling calendars
- 2048 flow queues (flow QCB) and 2048 associated scheduler control blocks (SCB)
- Target port queues
- Discard Queue
- Wrap Queue

6.4.2.1 Scheduling Calendars

The egress scheduler selects a flow queue to service every scheduler_tick. The duration of a scheduler_tick is determined by the configuration of the DRAM Parameter register 11/$\overline{10}$ field (bit 6 only). When set to 0, a scheduler_tick is 150 ns. When set to 1 a scheduler_tick is 165 ns.

There are three types of scheduling calendars used in the egress calendar design, time based, weighted fair queuing (WFQ), and wrap.

Time Based Calendars

The time based calendars are used for guaranteed bandwidth (LLS or NLS) and for peak bandwidth shaping.

Weighted Fair Queuing Calendars

The weighted fair queuing calendars allocate available bandwidth to competing flows on a per port basis. Available bandwidth is defined as the bandwidth left over after the flows in the NLS and LLS calendars get their bandwidth. A WFQ calendar is selected for service only when no service is required by the NLS, LLS or PBS calendars and the target port queue does not exceed a programmable threshold. The use of this threshold is the method that assures the WFQ calendar dequeues frames to the target port at a rate equal to the port's available bandwidth.

Wrap Calendar

The Wrap calendar is a 2-entry calendar for flows that use the wrap port.

Selection Algorithm Between Calendar Types

Selection between calendar types occurs each scheduler_tick using a fixed priority mechanism. The priority for each calendar type, with 1 highest, is:

1. Time based LLS
2. Time based NLS
3. Time based Peak bandwidth shaping
4. Weighted fair queuing
5. Wrap

6.4.2.2 Flow Queues

There are 2048 flow queues (flow QCBs) and scheduler control blocks. A flow QCB contains information about a single flow. Information that must be configured before the flow QCB can be used is listed here.

Configuring Sustainable Bandwidth (SSD)

The Sustained Service Rate (SSR) is defined as the minimum guaranteed bandwidth provided to the flow queue. It is implemented using either the LLS or NLS calendars. The following transform is used to convert Sustained Service Rate from typical bandwidth specifications to scheduler step units (SSD):

```
SSD = (512 (bytes) / scheduler_tick (sec) ) / Sustained_Service_Rate (bytes/sec)
```

The flow QCB SSD field is entered in exponential notation as X *16^{Y}, where X is the SSD.V field and Y is the SSD.E field . When a Sustained Service Rate is not specified, the flow QCB SSD field must be set to 0 during configuration.

Values of SSD that would cause the calendars to wrap should not be specified. Knowing the maximum frame size, SSD_{MAX} can be bound by:

$$SSD_{MAX} \leq 1.07 * 10E+9 / \text{Maximum Frame Size (bytes)}$$

An SSD value of 0 indicates that this flow queue has no defined guaranteed bandwidth component (SSR).

Configuring Flow QCB Flow Control Thresholds (TH)

When the number of bytes enqueued in the flow queue exceeds the Flow QCB Threshold (TH), the flow queue is considered to be congested. The flow control hardware uses this congestion indication to select the transmit probability. The following transform is used to specify this threshold:

TH.E	Threshold value (units in twin buffers)
0	TH.V * 2**6
1	TH.V * 2**9
2	TH.V * 2**12
3	TH.V * 2**15

TH.V and TH.E fields are components of the Flow Control Threshold field . The number of bytes in a twin

buffer is approximately 106 bytes. A TH value of 0 disables threshold checking.

Configuring Best Effort Service (QD)

The Queue weight is used to distribute available bandwidth among queues assigned to a port. The remaining available bandwidth on a port is distributed among contending queues in proportion to the flow's queue weight.

The following transform is used to convert the Queue Weight into scheduler step units (QD):

```
QD = 1/ (Queue Weight)
```

A QD of 0 indicates that the flow queue has no best effort component.

Configuring Peak Best Effort Bandwidth (PSD)

Peak Service Rate is defined as the additional bandwidth that this flow queue is allowed to use (the difference between the guaranteed and the peak bandwidth). For example, if a service level agreement provided for a guaranteed bandwidth of 8 Mbps and a peak bandwidth of 10 Mbps, then the Peak Service Rate is 2 Mbps. The following transform is used to convert Peak Service Rate from typical bandwidth specifications to scheduler step units (PSD):

```
PSD = (512 (bytes) / scheduler_tick (sec) ) / Peak_Service_Rate (bytes/sec)
```

The flow QCB PSD field is entered in exponential notation as $X * 16^{Y}$, where X is the PSD.V field and Y is the PSD.E field. A PSD value of 0 indicates that this flow queue has no Peak Service Rate component.

Target Port (TP)

The destination target port id.

Target Port Priority (P)

The Target Port Priority selects either the LLS or NLS calendar for the guaranteed bandwidth component of a flow queue. Values of 0 (high) and 1 (low) select the LLS or NLS calendar respectively.

During an enqueue to a target port queue, the P selects the correct queue. This field is also used in flow control.

Transmit Probability

Flow control uses transmit probability. The flow control algorithms running in picocode update this field periodically.

6.4.2.3 Target Port Queues

There are 82 target port queues, including 40 target ports with two priorities, a discard port, and a wrap port. The scheduler dequeues frames from flow QCBs and places them into the target port queue that is designated by the flow QCB's Target Port (TP) and priority (P) fields. A combination of a work conserving round robin and an absolute priority selection services target port queues. The round robin selects among the 40 media ports (target port ids 0 through 39) within each priority class (high and low as shown in *Figure 44* on page 142). A secondary selection using absolute priority is performed with the priority order as:

1. High Priority Target Port queues
2. Low Priority Target Port queues
3. Discard queue
4. Wrap queue

Information that must be configured for each target port queue is:

Port Queue Threshold (Th_PQ)

When the number of bytes enqueued in the target port queue exceeds the port queue threshold, the corresponding WFQ calendar cannot be selected for service. This back pressure to the WFQ calendars assures that best effort traffic is not allowed to fill the target port queue ahead of frames dequeued from the LLS and NLS calendars. The back pressure is the mechanism by which the target port bandwidth is reflected in the operation of the WFQ calendars.

The following transform is used to specify Th_PQ:

```
Th_PQ = Port_Queue_Threshold (bytes)/106 (bytes)
```

When configuring this threshold, it is recommended that the value used be larger than the MTU of the target port.

Port Queue + Flow Queue Threshold (Th_PQ+FQ)

This is threshold for the total number of bytes in the target port queue plus the total number of bytes in all flow queues that are configured for this target port. When this threshold is exceeded by the value in the queue, the target port is congested. The flow control mechanisms use this congestion indication to select a transmit probability.

The following transform is used to specify Th_PQ+FQ:

```
Th_PQ+FQ = Port_Queue+Scheduler_Threshold (bytes)/106 (bytes)
```

6.4.3 Configuring Flow Queues

Table 53 illustrates how to configure a flow QCB using the same set of combinations found in *Table 52: Valid Combinations of Scheduler Parameters* on page 143.

Table 53: Configure a Flow QCB

QoS	QCB.P	QCB.SD	QCB.PSD	QCB.QD
Low latency with Guaranteed BW shaping	0	≠0	0	0
Normal latency with Guaranteed BW shaping Best Effort	1	≠0	0	0
Best Effort	1	0	0	≠0
Best Effort with Peak Rate	1	0	≠0	≠0
Normal latency with Guaranteed BW shaping with Best Effort	1	≠0	0	≠0
Normal latency with Guaranteed BW shaping with Best Effort, and Peak Rate	1	≠0	≠0	≠0

6.4.3.1 Additional Configuration Notes

1. Once the scheduler is enabled via the Memory Configuration Register, the picocode is unable to enqueue directly to a target port queue. To disable the scheduler, the software system design must assure that the scheduler is drained of all traffic. All target port queue counts can be examined via the CAB; when all counts are zero, the scheduler is drained.
2. A flow QCB must be configured for discards (TP=41). A Sustained service rate must be defined (SSD ≠ 0). A Peak Service rate and Queue weight must not be specified (PSD=0 and QD=0).
3. Two flow QCBs must be defined for wrap traffic. One flow QCB must be defined for Guided traffic (TP = 40) and one for frame traffic (TP = 42). For these QCBs, the Peak Service rate and Sustained service rate must not be specified (PSD=0 and SSD=0). Queue weight must be non-zero (QD ≠ 0).

7. Embedded Processor Complex

7.1 Overview

The Embedded Processor Complex (EPC) provides and controls the programmability of the NP4GS3. The EPC consists of the following components:

- Eight Dyadic Protocol Processors Units (DPPU)

  Each DPPU consists of two Core Language Processors (CLP), eight shared coprocessors, one coprocessors databus, one coprocessor command bus, and a shared memory pool. Each CLP contains one ALU and supports two picocode threads, so each DPPU has four threads (see section *7.1.1 Thread Types* on page 152 for more information). Although there are 32 independent threads, each CLP can execute the command of only one of its picocode threads, so at any one instant in time only 16 threads are executing on all of the CLPs. The CLPs and coprocessors contain independent copies of each thread's registers and arrays. Most coprocessors perform specialized functions as described below and can operate concurrently with each other and with the CLPs.

  - Tree Search Engine (TSE)

    The TSE coprocessor is part of the DPPUs. It has commands for tree management, direct access to the Control Store (CS), and search algorithms such as Full Match (FM), Longest Prefix Match (LPM), and Software Managed (SMT) trees. For more information, see *Section 8* on page 219.

- Interrupts and Timers

  The NP4GS3 has four Interrupt Vectors. Each interrupt can be configured to initiate a dispatch to occur to one of the threads for processing.

  There are four timers that can be used to generate periodic interrupts.

- Instruction Memory

  The Instruction Memory consists of eight embedded RAMs that are loaded during initialization and contain the picocode for forwarding frames and managing the system. The total size is 16 K instructions. The memory is 4-way interleaved with four RAMs for the first 8 K and four RAMS for the remaining 8 K.

- Control Store Arbiter (CSA)

  The CSA controls access to the Control Store (CS) which allocates memory bandwidth among the threads of all the dyadic protocol processors. The CS is shared among the tree search engines and the picocode can directly access the CS through commands to the TSE coprocessor. The TSE coprocessor also accesses the CS during tree searches.

- Dispatch Unit

  The Dispatch Unit dequeues frame information from the Ingress-EDS and Egress-EDS queues. After dequeue, the Dispatch Unit reads part of the frame from the Ingress or Egress Data Store (DS) and places it into the DataPool. As soon as a thread becomes idle, the Dispatch Unit passes the frame with appropriate control information, to the thread for processing.

  The Dispatch Unit also handles timers and interrupts by dispatching the work required for these to an available thread.

- Completion Unit (CU)

  The CU performs two functions:

  - It provides the interfaces between the EPC and the Ingress and Egress EDSs. Each EDS performs an enqueue action whereby a frame address, together with appropriate parameters, is queued in a transmission queue or a Dispatch Unit queue.
  - The CU guarantees frame sequence. Since multiple threads can process frames belonging to the same flow, the CU ensures that all frames are enqueued in the Ingress or Egress transmission queues in the proper order.

- Hardware Classifier (HC)

  The HC provides hardware assisted parsing of frame data that is dispatched to a thread. The results are used to precondition the state of a thread by initializing the thread's General Purpose and Coprocessor Scalar Registers and a starting instruction address for the Core Language Processor. Parsing results indicate the type of Layer 2 encapsulation, as well as some information about the Layer 3 packet. Recognizable Layer 2 encapsulations include PPP, 802.3, DIX V2, LLC, SNAP header, and VLAN tagging. Reportable Layer 3 information includes IP and IPx network protocols, five programmable network protocols, the detection of IP option fields, and transport protocols (UDP, TCP) for IP.

- Ingress and Egress Data Store Interface and Arbiter

  Each thread has access to the Ingress and Egress Data Store through a Data Store Coprocessor. Read access is provided when reading "more Data" and write access is provided when writing back the contents of the Shared Memory Pool (SMP) to the Data Store. One Arbiter is required for each data store since only one thread at a time can access either data store.

- Control Access Bus (CAB) Arbiter

  Each thread has access to the CAB, which permits access to all memory and registers in the network processor. The CAB Arbiter arbitrates among the threads for access to the CAB.

- Debugging and Single Step Control

  The CAB allows the GFH thread to control each thread on the chip for debugging purposes. For example, the CAB can be used by the GFH thread to run a selected thread in single-step execution mode. (See *Section 12. Debug Facilities* on page 337 for more information.)

- Policy Manager

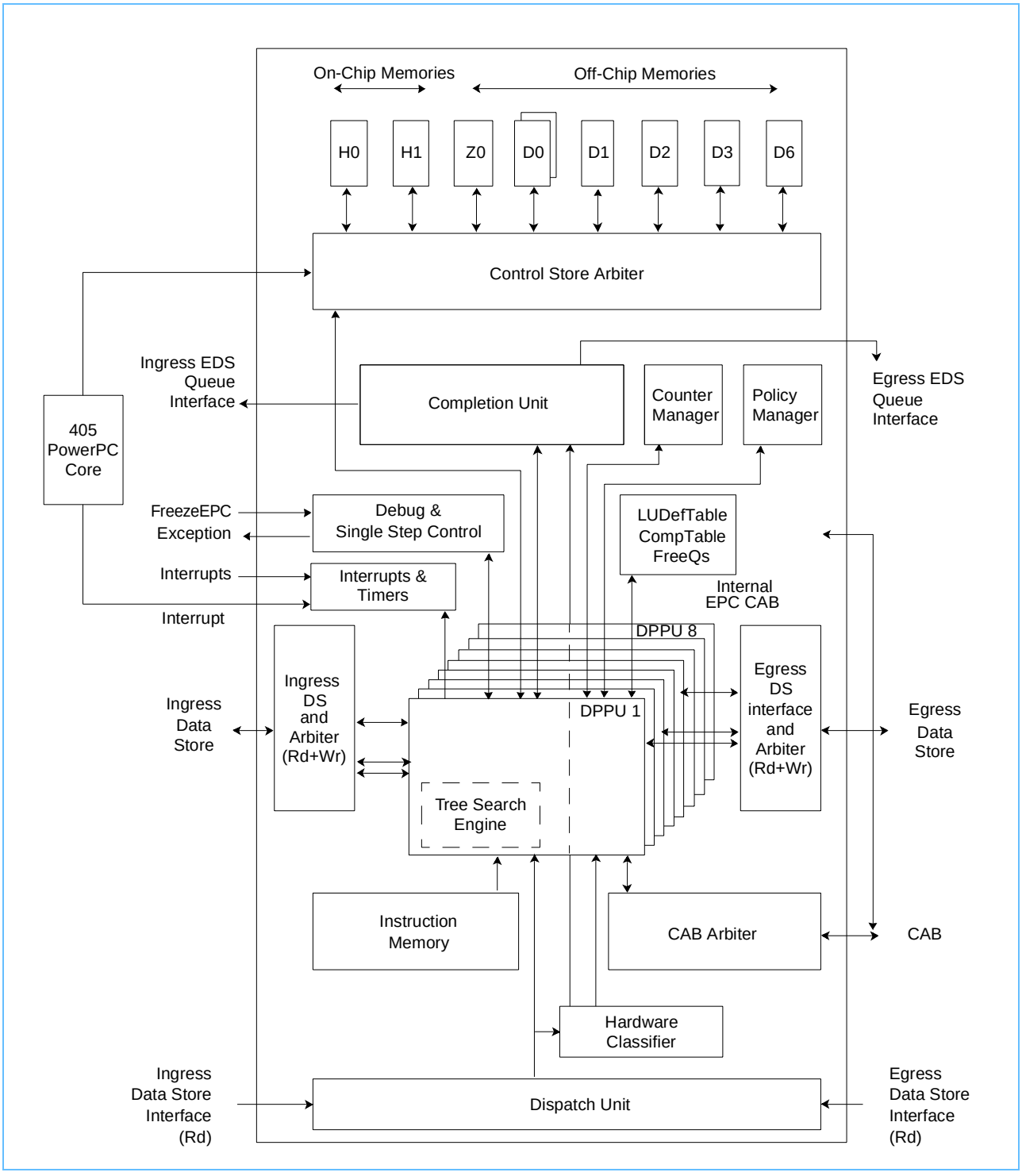
  The Policy Manager is a hardware assist of the EPC that performs policy management on up to 1 K ingress flows. It supports four management algorithms. One algorithm pair is "Single Rate Three Color Marker," and the other is "Two Rate Three Color Marker." Both are operated in color-blind or color-aware mode. The algorithms are specified in IETF RFCs 2697 and 2698 (available at http://www.ietf.org).

- Counter Manager

  The Counter Manager is a hardware assist engine used by the EPC to manage counters defined by the picocode for statistics, flow control, and policy management. The Counter Manager is responsible for counter updates, reads, clears, and writes. The Counter Manager's interface to the Counter Coprocessor provides picocode access to these functions.

- LuDefTable, CompTable, and Free Queues are tables and queues for use by the tree search engine. (For more information, see *8.2.5.1 The LUDefTable* on page 233 and the IBM PowerNP NP4GS3 Hardware Reference Manual.)

Figure 45: Embedded Processor Complex Block Diagram



7.1.1 Thread Types

The EPC has 32 threads that can simultaneously process 32 frames. A thread has a unique set of General Purpose, Scalar, and Array registers, but shares execution resources in the CLP with another thread and execution resources in the coprocessors with three other threads. Threads can be classified into five different categories:

- General Data Handler (GDH)

  There are 28 GDH threads. GDHs are used for forwarding frames.

- Guided Frame Handler (GFH)

  There is one GFH thread available in the EPC. A guided frame can only be processed by the GFH thread, but the GFH can be configured to process data frames like a GDH thread. The GFH executes guided frame related picocode, runs chip management related picocode, and exchanges control information with a control point function or a remote network processor. When there is no such task to perform and the option is enabled, the GFH may execute frame forwarding related picocode.

- General Table Handler (GTH)

  There is one GTH thread available in the EPC. The GTH executes tree management commands not available to other threads. The GTH performs actions including hardware assist to perform tree inserts, tree deletes, tree aging, and rope management. The GTH can process data frames like a GDH when there are no tree management functions to perform.

- General PowerPC Handler Request (GPH-Req)

  There is one GPH-Req thread available in the EPC. The GPH-Req thread processes frames bound to the embedded PowerPC. Work for this thread is the result of a re-enqueue action from another thread in the EPC to the GPQ queue. The GPH-Req thread moves data bound for the PowerPC to the PowerPC's Mailbox (a memory area) and then notifies the PowerPC that it has data to process. (See section *10.8 Mailbox Communications and DRAM Interface Macro on page 291* for more information.)

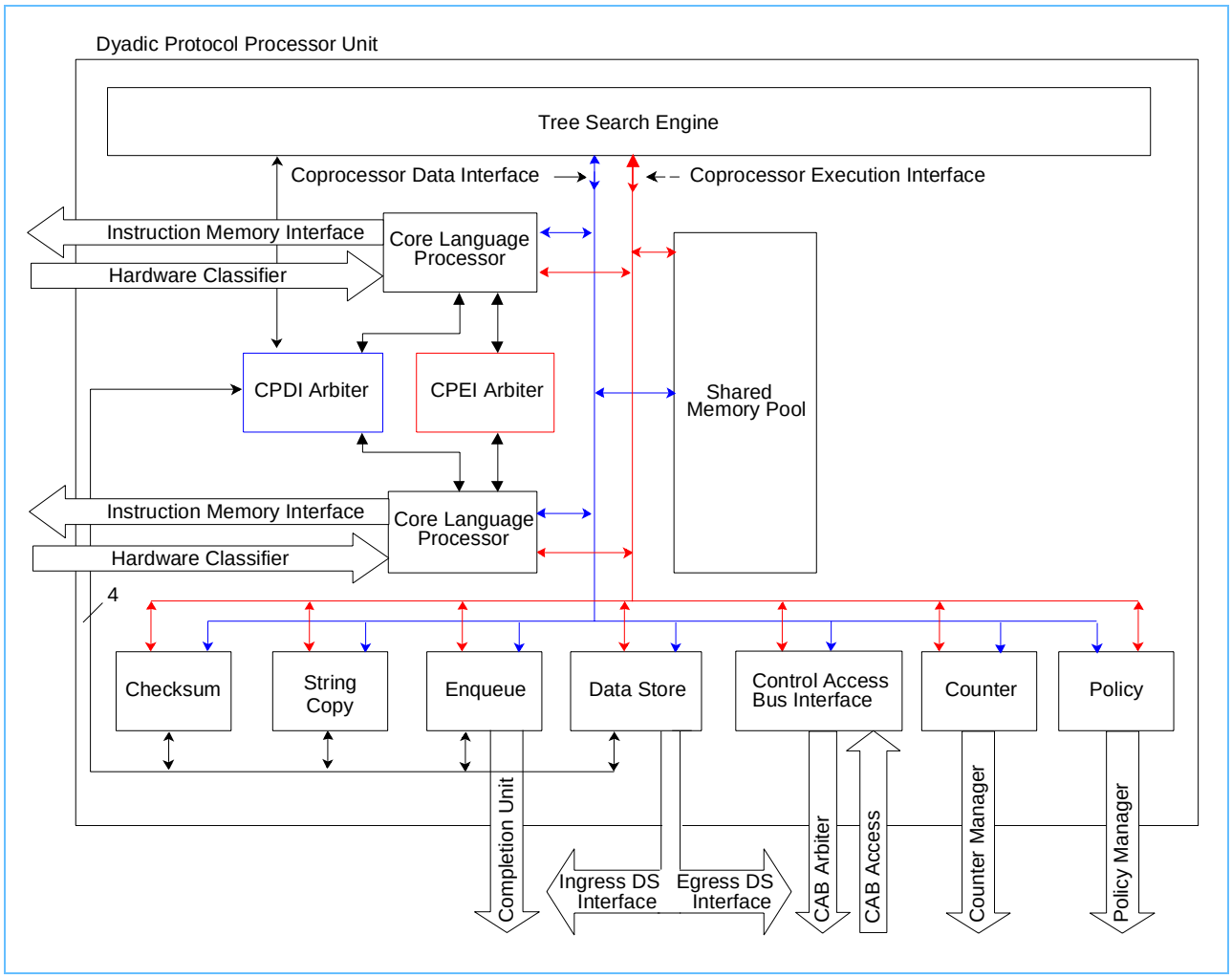
- General PowerPC Handler Response (GPH-Resp)

  There is one GPH-Resp thread available in the EPC. The GPH-Resp thread processes responses from the embedded PowerPC. Work for this thread is dispatched due to an interrupt initiated by the PowerPC and does not use Dispatch Unit memory. All the information used by this thread is found in the embedded PowerPC's Mailbox. (See section *10.8 Mailbox Communications and DRAM Interface Macro on page 291* for more information.)

7.2 Dyadic Protocol Processor Unit (DPPU)

Each DPPU consists of two Core Language Processors, eight coprocessors, one coprocessors databus, one coprocessor command bus, (the Coprocessor Databus and Coprocessor Execution interfaces), and a 4 K byte shared memory pool (1 K bytes per thread). Each CLP supports two threads, so each DPPU has four threads which execute the picocode that is used to forward frames, update tables, and maintain the network processor. The first DPPU contains the GFH, GTH, and the PPC threads and the other seven DPPUs contain the GDH threads. Each DPPU interfaces to the Instruction Memory, the Dispatch Unit, the Control Store Arbiter, the Completion Unit, the Hardware Classifier, the interface and arbiter to the Ingress and Egress Data Stores, the CAB Arbiter, the debugging facilities, the Counter Manager, the Policy Manager, and the LuDefTable and Comp Free Queues.

Figure 46: Dyadic Protocol Processor Unit Block Diagram

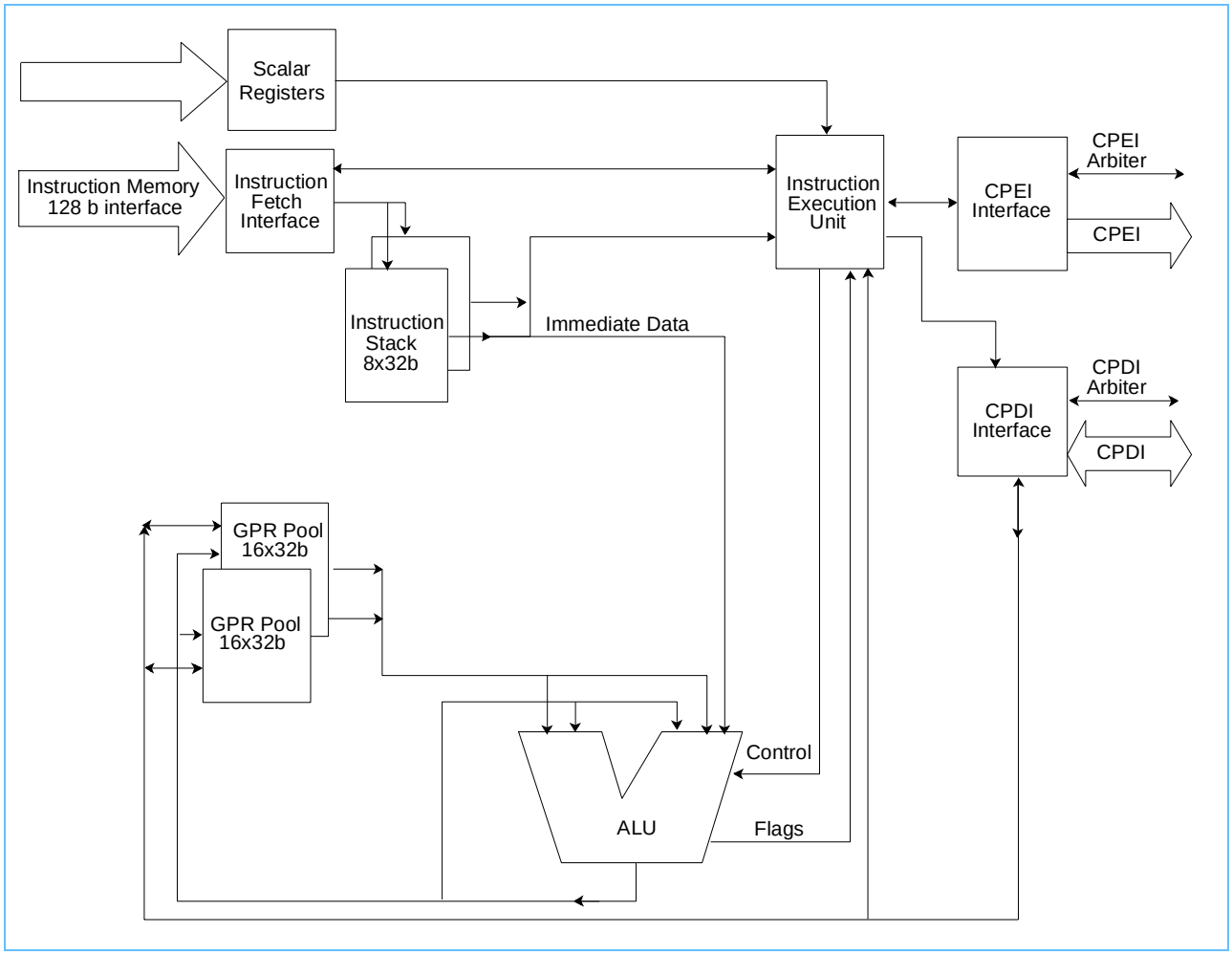


7.2.1 Core Language Processor (CLP)

Each DPPU contains two CLPs. The CLP executes the EPC's core instruction set and controls thread swapping and instruction fetching. Each CLP is a 32-bit picoprocessor consisting of:

- 16 32-bit or 32 16-bit General Purpose Registers (GPR) per thread. (For more information, see *Table 54: Core Language Processor Address Map on page 156* .)
- A one-cycle ALU supporting an instruction set that includes:
  - Binary addition and subtraction
  - Bit-wise Logical AND, OR, and NOT
  - Compare
  - Count leading zeros
  - Shift left and right Logical
  - Shift right arithmetic
  - Rotate Left and Right
  - Bit manipulation commands: set, clear, test, and flip
  - GPR transfer of Halfword to Halfword, Word to Word, and Halfword to Word with and without sign extensions.
  - All instructions can be coded to run conditionally. This eliminates the need for traditional branch-and-test coding techniques, improving performance and reducing the size of the code. All arithmetic and logical instructions can be coded to execute without setting ALU status flags.
- Management for handling two threads with zero overhead for context switching
- Read-only scalar registers that provide access to interrupt vectors, timestamp, output of a pseudo random number generator, picoprocessor status, work queue status (such as the ingress and egress data queues), and configurable identifiers (such as the blade identification). (For more information, see *Table 54: Core Language Processor Address Map on page 156* .)
- 16-word instruction prefetch shared by each thread
- Instruction Execution unit that executes branch instructions, instruction fetch, and access to the coprocessors
- Coprocessor Data Interface (CPDI):
  - Access from any byte, halfword, or word of a GPR to an array, or from an array to a GPR.
  - Access to coprocessor scalar registers
  - Various sign, zero, and one extension formats
  - Quadword transfers within the coprocessor arrays
- Coprocessor Execution Interface (CPEI):
  - Synchronous or asynchronous coprocessor operation
  - Multiple coprocessor synchronization
  - Synchronization and Branch-on-Coprocessor Return Code

Figure 47: Core Language Processor



7.2.1.1 Core Language Processor Address Map

Table 54: Core Language Processor Address Map

Name	Register (Array)[1] Number	Size (Bits)	Access	Description
PC	x‘00’	16	R	Program Counter. Address of the next instruction to be executed.
ALUStatus	x‘01’	4	R	The current ALU status flags: 3 Zero 2 Carry 1 Sign 0 Overflow
LinkReg	x‘02’	16	R/W	Link Register. Return address for the most recent subroutine
CoPStatus	x‘03’	10	R	Indicates whether the coprocessor is busy or idle. A coprocessor that is Busy will stall the CLP when the coprocessor command was executed synchronously or a wait command was issued for the coprocessor. 1 Coprocessor is Busy 0 Coprocessor is Idle
CoPRtnCode	x‘04’	10	R	The definition of OK/KO is defined by the coprocessor. 1 OK 0 Not OK
ThreadNum	x‘05’	5	R	The thread number (0..31)
TimeStamp	x‘80’	32	R	Free-running, 1 ms timer
RandomNum	x‘81’	32	R	Random number for programmer’s use
IntVector0	x‘83’	32	R	Read-Only copy of Interrupt Vector 0. Reading this register has no effect on the actual Interrupt Vector 0.
IntVector1	x‘84’	32	R	Read-Only copy of Interrupt Vector 1. Reading this register has no effect on the actual Interrupt Vector 1.
IntVector2	x‘85’	32	R	Read-Only copy of Interrupt Vector 2. Reading this register has no effect on the actual Interrupt Vector 2.
IntVector3	x‘86’	32	R	Read-Only copy of Interrupt Vector 3. Reading this register has no effect on the actual Interrupt Vector 3.
IdleThreads	x‘87’	32	R	Indicates that a thread is enabled and idle
QValid	x‘88’	32	R	Indicates status of the queues (valid or invalid).
My_TB	x‘89’	6	R	My Target Blade. The Blade Number representing the blade in which this Networking Processor chip is currently residing (see *13.13.9 My Target Blade Address Register (My_TB)* on page 365)
SW_Defined_A	x‘8A’	32	R	Software Defined Register
SW_Defined_B	x‘8B’	32	R	Software Defined Register
SW_Defined_C	x‘8C’	32	R	Software Defined Register
Version_ID	x‘8F’	32	R	Contains the version number of the hardware.
GPRW0	x‘C0’	32	R	General Purpose Register W0
GPRW2	x‘C1’	32	R	General Purpose Register W2
GPRW4	x‘C2’	32	R	General Purpose Register W4
GPRW6	x‘C3’	32	R	General Purpose Register W6

1. A number in parentheses is the array number for this coprocessor. Each array has a register number and an array number.

Table 54: Core Language Processor Address Map (Continued)

Name	Register (Array)[1] Number	Size (Bits)	Access	Description
GPRW8	x'C4'	32	R	General Purpose Register W8
GPRW10	x'C5'	32	R	General Purpose Register W10
GPRW12	x'C6'	32	R	General Purpose Register W12
GPRW14	x'C7'	32	R	General Purpose Register W14
GPRW16	x'C8'	32	R	General Purpose Register W16
GPRW18	x'C9'	32	R	General Purpose Register W18
GPRW20	x'CA'	32	R	General Purpose Register W20
GPRW22	x'CB'	32	R	General Purpose Register W22
GPRW24	x'CC'	32	R	General Purpose Register W24
GPRW26	x'CD'	32	R	General Purpose Register W26
GPRW28	x'CE'	32	R	General Purpose Register W28
GPRW30	x'CF'	32	R	General Purpose Register W30
PgramStack0	x'FC' (0)	128	R/W	Entries in the Program Stack (used by the CLP hardware to build instruction address stacks for the branch and link commands)
PgramStack1	x'FD' (1)	128	R/W	Entries in the Program Stack (used by the CLP hardware to build instruction address stacks for the branch and link commands)

1. A number in parentheses is the array number for this coprocessor. Each array has a register number and an array number.

7.2.2 DPPU Coprocessors

Each DPPU coprocessor is a specialized hardware assist engine that runs in parallel with the two CLPs and performs functions that would otherwise have required a large amount of serialized picocode. The functions include modifying IP headers, maintaining flow information used in flow control algorithms, accessing internal registers via the CAB, maintaining counts for flow control and for standard and proprietary Management Information Blocks (MIB), and enqueueing frames to be forwarded.

The DPPU coprocessors are:

- Data Store
- Tree Search Engine (see *Section 8* on page 219)
- Control Access Bus Interface
- Enqueue
- Checksum
- String Copy
- Policy
- Counter

A thread's address space is a distributed model in which registers and arrays reside within the coprocessors (each coprocessor maintains resources for four threads and, for address mapping purposes, the CLP is considered to be a coprocessor). Each coprocessor can have a maximum of 252 Scalar Registers and four Arrays. The address of a Scalar Register or Array within the DPPU becomes a combination of the coprocessor number and the address of the entity within the coprocessor.

Likewise, the coprocessor instruction is a combination of the coprocessor number and the coprocessor opcode.

The EPC coprocessors are numbered as shown in *Table 55*. The number is used in accesses on both the Coprocessor Execute and Data Interfaces. The TSE is mapped to two coprocessor locations so that a thread can execute two searches simultaneously.

Table 55: Coprocessor Instruction Format

Coprocessor Number	Coprocessor
0	Core Language Processor (CLP)
1	Data Store Interface
2	Tree Search Engine 0
3	Tree Search Engine 1
4	CAB Interface
5	Enqueue
6	Checksum
7	String Copy
8	Policy
9	Counter

7.2.3 The Data Store Coprocessor

The Data Store Coprocessor provides an interface between the EPC and the Ingress Data Store, which contains frames that have been received from the media, and the Egress Data Store, which contains reassembled frames received from the switch interface. The Data Store Coprocessor also receives configuration information during the dispatch of a timer event or interrupt.

There is one set of scalar registers and arrays defined for each thread supported by the Data Store Coprocessor. The scalar registers control the accesses between the shared memory pool and the ingress and egress data stores. They are maintained in the Data Store Coprocessor. The arrays are defined in the Shared Memory Pool (see *7.2.10 Shared Memory Pool* on page 198):

- The DataPool, which can hold eight quadwords
- Two Scratch Memory Arrays, which hold eight and four quadwords respectively
- The Configuration Quadword Array, which holds the port configuration data that was dispatched to the thread.

These Shared Memory Pool arrays can be conceptualized as a work area for the Data Store Coprocessor: instead of reading or writing small increments (anything less than a quadword, or 16 bytes) directly to a Data Store, a larger amount (1 to 4 quadwords per operation) of frame data is read from the Data Store into these Shared Memory Pool arrays or from the these arrays into the Data Store.

The Data Store Coprocessor has nine commands available to it. For more information, see section *7.2.3.2 Data Store Coprocessor Commands* on page 165.

7.2.3.1 Data Store Coprocessor Address Map

Table 56: Data Store Coprocessor Address Map (A thread's scalar registers and arrays that are mapped within the Data Store coprocessor)

Name	Register (Array)[1] Number	Size (bits)	Access	Description
DSA	x'00'	19	R/W	Address of Ingress or Egress Data Store. Used in all commands except "read more" and "dirty". (Ingress uses the least significant 11 bits and Egress uses all 19 bits for the address.)
LMA	x'01'	6	R/W	Quadword address of Shared Memory Pool. Used in all commands except "read more". (See *7.2.10 Shared Memory Pool* on page 198.)
CCTA	x'02'	19	R/W	Current address of the ingress cell or the egress twin that was dispatched to the thread into the datapool. Used in "read more" and "dirty" commands. The value is unitized at dispatch and updated on "read more" commands that pass though cell or twin boundaries.
NQWA	x'03'	3	R/W	Quadword address of the location in the DataPool where the next quadword will be written.
iProtocolType	x'04'	16	R	Layer 3 Protocol identifier set by the Hardware Classifier. (See section *7.4 Hardware Classifier* on page 202)
DirtyQW	x'05'	8	R/W	Quadwords in the DataPool that have been written and therefore may not be equivalent to the corresponding Data Store data. Used by the dirty update commands to write back any modified data into the corresponding Data Store.

1. A number in parentheses is the array number for this coprocessor. Each array has a register number and an array number.

Table 56: Data Store Coprocessor Address Map (A thread's scalar registers and arrays that are mapped within the Data Store coprocessor)

Name	Register (Array)[1] Number	Size (bits)	Access	Description
BCI2Byte	x'06'	14/20	R/W	A special purpose register. The picocode running in the CLP writes a 20-bit BCI value, but when the register is read, it returns the 14-bit byte count represented by the BCI.
Disp_DSU	x'07'	2	R/W	Initialized during dispatch to contain the same value as the DSU field in the Egress FCBPage. This register is used by Egress write commands to determine which Egress Data Store the Data Store coprocessor should access when writing data. This register has no meaning for Ingress frames.
Disp_DSUSel	x'08'	1	R/W	Initialized during dispatch to contain the same value as the DSU_Sel field in the Egress FCBPage. This register is used by Egress read commands to determine which Egress Data Store the Data Store coprocessor should access when reading data. This register has no meaning for Ingress frames.
Disp_Ingress	x'09'	1	R	Dispatched frame's type 0 Egress frame 1 Ingress frame
ConfigQW	x'FC' (0)	128	R/W	Port Configuration Table Entry for this frame
ScratchMem0	x'FD' (1)	512	R/W	User Defined Array (to use to store temporary information, build new frames, and so on.)
ScratchMem1	x'FE' (2)	1024	R/W	User Defined Array (to use to store temporary information, build new frames, and so on.)
DataPool	x'FF' (3)	1024	R/W	Contains frame data from the Dispatch Unit

1. A number in parentheses is the array number for this coprocessor. Each array has a register number and an array number.

The DataPool and Data Store Access

Upon frame dispatch, the Dispatch Unit automatically copies the first N quadwords of a frame from the Data Store into the first N quadword positions of the DataPool. The value of N is programmable in the Port Configuration Memory. Typically, values of N are as follows:

N = 4	Ingress frame dispatch
N = 2	Egress unicast frame dispatch
N = 4	Egress multicast frame dispatch
N = 0	Interrupts and timers

The read more commands (RDMOREI and RDMOREE) assist the picocode's reading of additional bytes of a frame by automatically reading the frame data into the DataPool at the next quadword address and wrapping automatically to quadword 0 when the boundary of the DataPool is reached. The picocode can also read or write the Ingress and Egress Data Stores at an absolute address, independent of reading sequential data after a dispatch.

For an Ingress Dispatch, the N quadwords are stored in the DataPool at quadword-address 0, 1, … N-1. Each quadword contains raw frame-bytes, (there are no cell header or frame headers for ingress frames in the DataPool). After an Ingress Dispatch, the DataPool contains the first N*16 bytes of the frame, where the first byte of the frame has byte-address 0. The Ingress DataPool Byte Address Definitions are listed in *Table 57*.

Table 57: Ingress DataPool Byte Address Definitions

Quadword	Byte Address															
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127

When reading more than N quadwords of a frame (using the RDMOREI command), the hardware automatically walks the Data Store's BCB chain as required. Quadwords read from the Data Store are written to the DataPool at consecutive quadword-locations, starting at quadword address N (where N is the number of quadwords written to the DataPool by the Dispatch Unit during frame dispatch). The quadword-address wraps from 7 - 0, therefore the picocode must save quadword 0 in case it is required later. (For example, the quadword can be copied to a scratch array).

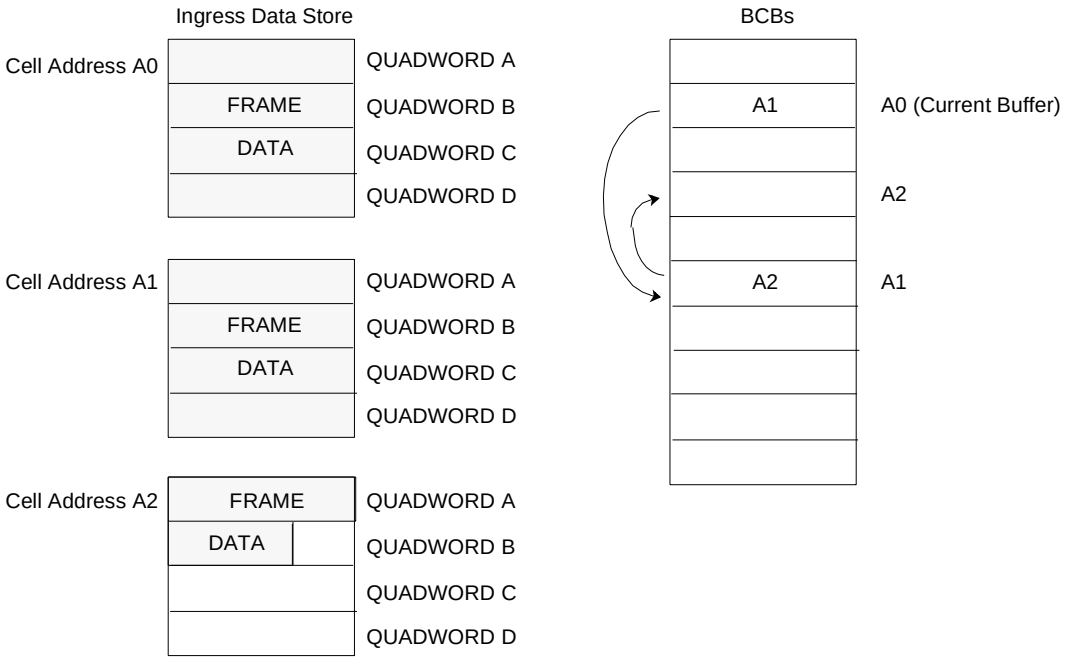
The Ingress Data Store can also be written and read with an absolute address. The address is a BCB address.

Reading from an absolute address can be used for debugging and to inspect the contents of the Ingress DS.

For absolute Ingress Data Store access (reads and writes), the picocode must provide the address in the Ingress Data Store, the quadword address in the DataPool, the number of quadwords to be transferred.

An example of a frame stored in the Ingress Data Store is shown in *Figure 48*:

Figure 48: A Frame in the Ingress Data Store



For an Egress Dispatch, the N quadwords are stored in the DataPool at a starting quadword-address that is determined by where the frame header starts in the twin, which in turn depends on how the frame is packed in the Switch cell and stored in the Egress Data Store (see *Figure 49*). GPR R0 is initialized during dispatch according to *Table 58: Egress Frames DataPool Quadword Addresses* on page 164. GPR R0 can be used as an index into the DataPool so that the variability of the location of the start of the frame in the datapool is transparent to the picocode.

Figure 49: A Frame in the Egress Data Store (Illustrating the effects of different starting locations)

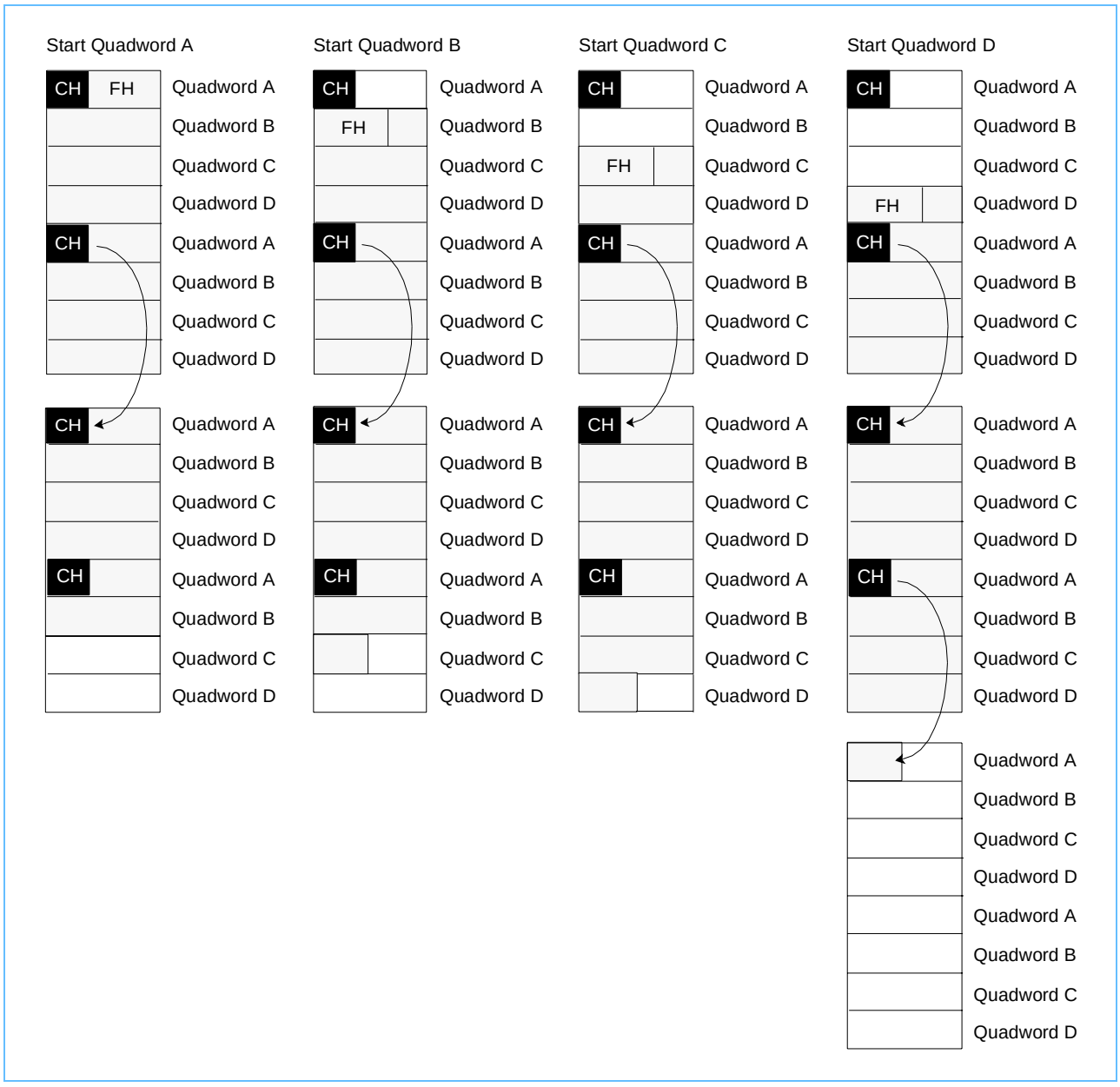


Table 58: Egress Frames DataPool Quadword Addresses

Frame Start In Twin Buffer	Frame Quadword Address in the DataPool	GPR R0
Frame starts at Quadword A in the twin buffer	0	0
Frame starts at Quadword B in the twin buffer	1	10
Frame starts at Quadword C in the twin buffer	2	26
Frame starts at Quadword D in the twin buffer	3	42

The relationship between quadwords A, B, C, and D in the twin buffer and the location of the quadword in the DataPool is always maintained. That is, Quadword A is always stored at quadword address 0 or 4, Quadword B is always stored at quadword address 1 or 5, Quadword C is always stored at quadword address 2 or 6, and Quadword D is always stored at quadword address 3 or 7.

In contrast with the Ingress, the DataPool for the Egress contains cell headers. When the exact content of the twin-buffer is copied into the DataPool, it may include the 6-byte NP4GS3 cell header if the quadword being copied comes from Quadword A in the twin buffer.

The DataPool can be accessed in two modes:

- Normal DataPool access:
  Accesses all bytes in the DataPool including the 6-byte cell header. For example, it can be used for guided frames, where information in the cell header may be important, or for debugging and diagnostics.
- Cell Header Skip DataPool access:
  Automatically skips the cell header from the DataPool. The hardware assumes a cell header to be present at quadword-address 0 and 4. For example, accessing DataPool[2] accesses the byte with physical address 8, DataPool[115] accesses the byte with physical address 127, and DataPool[118] also accesses the byte with physical address 8. The maximum index that can be used for this access mode is 231. This mode is shown in *Table 59*.

Table 59: DataPool Byte Addressing with Cell Header Skip

Quadword	Byte Address															
0	—	—	—	—	—	—	0 116	1 117	2 118	3 119	4 120	5 121	6 122	7 123	8 124	9 125
1	10 126	11 127	12 128	13 129	14 130	15 131	16 132	17 133	18 134	19 135	20 136	21 137	22 138	23 139	24 140	25 141
2	26 142	27 143	28 144	29 145	30 146	31 147	32 148	33 149	34 150	35 151	36 152	37 153	38 154	39 155	40 156	41 157
3	42 158	43 159	44 160	45 161	46 162	47 163	48 164	49 165	50 166	51 167	52 168	53 169	54 170	55 171	56 172	57 173
4	—	—	—	—	—	—	58 174	59 175	60 176	61 177	62 178	63 179	64 180	65 181	66 182	67 183
5	68 184	69 185	70 186	71 187	72 188	73 189	74 190	75 191	76 192	77 193	78 194	79 195	80 196	81 197	82 198	83 199
6	84 200	85 201	86 202	87 203	88 204	89 205	90 206	91 207	92 208	93 209	94 210	95 211	96 212	97 213	98 214	99 215
7	100 216	101 217	102 218	103 219	104 220	105 221	106 222	107 223	108 224	109 225	110 226	111 227	112 228	113 229	114 230	115 231

Fewer frame-bytes may be available in the DataPool for the Egress due to the presence of cell headers. *Table 60* shows the number of frame-bytes in the DataPool after a frame dispatch.

Table 60: Number of Frame-bytes in the DataPool

Number Of Quadwords Read	Start Quadword A	Start Quadword B	Start Quadword C	Start Quadword D	Guaranteed
1	10	16	16	16	10
2	26	32	32	26	26
3	42	48	42	42	42
4	58	58	58	58	58
5	68	74	74	74	68
6	84	90	90	84	84
7	100	106	100	100	100
8	116	116	116	116	116

For example, when 24 bytes of frame data are always needed, the Port Configuration Memory must be programmed with the N (number of quadwords to dispatch) equal to two. When 32 bytes are always needed, the number of quadwords to dispatch must be set to three.

After a dispatch, picocode can use the RDMOREE command when more frame-data is required. One, two, three, or four quadwords can be requested. Consult the “Guaranteed” column in *Table 60* to translate the necessary number of bytes into the greater number of quadwords that must be read. For example, if the picocode must dig into the frame up to byte 64, five quadwords are required. In this example, the number of quadwords specified with the RDMOREE command equals 5-N, where N is the number of quadwords initially written in the DataPool by the Dispatch Unit.

A general rule: each set of four quadwords provide exactly 58 bytes.

7.2.3.2 Data Store Coprocessor Commands

The Data Store Coprocessor provides the following commands:

WREDS	Write Egress Data Store. Allows the CLP to read data from one of the arrays in the Shared Memory Pool (DataPool and Scratch Memory Array) and write it to the Egress Data Store (in multiples of quadwords only).
RDEDS	Read Egress Data Store. Allows the CLP to read data from the Egress Data Store and write it into one of the arrays in the Shared Memory Pool (in multiples of quadwords only).
WRIDS	Write Ingress Data Store. Allows the CLP to write data to the Ingress data store (in multiples of quadwords only).
RDIDS	Read Ingress Data Store. Allows the CLP to read data from the Ingress Data Store (in multiples of quadwords only).

RDMOREE	Read More Frame Data from the Egress Data Store. A hardware assisted read from the Egress Data Store. RDMOREE continues reading the frame from where the dispatch or last “read more” command left off and places the data into the DataPool. As data is moved into the DataPool, the hardware tracks the current location in the frame that is being read and captures the link pointer from the twin buffers in order to determine the address of the next twin buffer. This address is used by the hardware for subsequent RDMOREE requests until the twin is exhausted and the next twin is read. Since the contents of the DataPool is a map of a twin's content, there is a potential for the frame data to wrap within the DataPool; the picocode keeps track of the data’s location within the DataPool.
RDMOREI	Read More Frame Data from the Ingress Data Store. A hardware assisted read from the Ingress Data Store. RDMOREI continues reading the frame from where the dispatch or last “read more” command left off and places the data into the DataPool. As data is moved into the DataPool, the hardware tracks the current location in the frame that is being read and captures the link maintained in the buffer control block area in order to determine the address of the frame’s next data buffer. This address is used by the hardware for subsequent RDMOREI requests until the data buffer is exhausted and the next buffer is read. The picocode keeps track of the frame data’s location within the DataPool.
LEASETWIN	Lease Twin Buffer. Returns the address of a free twin buffer (used when creating new data in the Egress Data Store).
EDIRTY	Update Dirty Quadword Egress Data Store. The coprocessor keeps track of the quadwords within the current twin that have been modified in the DataPool array. EDIRTY allows the CLP to write only the “dirty” data back to the Egress data store (in multiples of quadwords only). This command is only valid within the DataPool array and for the buffer represented by the scalar register Current Cell/Twin Address.
IDIRTY	Update Dirty Quadword Ingress Data Store. The coprocessor keeps track of the quadwords within the current buffer that have been modified in the DataPool array. IDIRTY allows the CLP to write only the “dirty” data back to the Ingress Data Store (in multiples of quadword units only). This command is only valid within the DataPool array and for the buffer represented by the scalar register Current Cell/ Twin Address.

Table 61: Data Store Coprocessor Commands Summary

Opcode	Command	Detail Section
0	WREDS	*WREDS (Write Egress Data Store) Command* on page 167
1	RDEDS	*RDEDS (Read Egress Data Store) Command* on page 168
2	WRIDS	*WRIDS (Write Ingress Data Store) Command* on page 168
3	RDIDS	*RDIDS (Read Ingress Data Store) Command* on page 169
5	RDMOREE	*RDMOREE (Read More Quadword From Egress) Command* on page 171
7	RDMOREI	*RDMOREI (Read More Quadword From Ingress) Command* on page 170
8	LEASETWIN	*LEASETWIN Command* on page 173
10	EDIRTY	*EDIRTY (Update Egress Dirty Quadwords) Command* on page 171
12	IDIRTY	*IDIRTY (Update Ingress Dirty Quadwords) Command* on page 172

Note: The Egress Data Store has a longer access time than the Ingress Data Store. For performance reasons it is recommended to do as much frame parsing on the Ingress as possible.

WREDS (Write Egress Data Store) Command

WREDS writes to an absolute address in the Egress Data Store. It can be specified if the quadwords are written to DS0, DS1, both DS0 and DS1, or if the decision is made automatically by information in the Dispatch DSU register.

Table 62: WREDS Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be written: 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
DSControl	2	Imm16(3..2)	Imm12(3..2)	Defines if the quadwords are written to DS0, DS1 or both: 00 Writes to the default DSU 01 Writes to DS0 10 Writes to DS1 11 Writes to DS0 and DS1
Disp_DSU	2	R		The default DSU, initialized at Dispatch
DSA	19	R		Data Store Address. The target twin address in the Egress Data Stores. The starting QW destination within the twin is determined by the 3 low-order bits of the LMA. 000 - 011 are QW 0-3 of the first buffer of the twin. 100-111 are QW 0-3 of the second buffer of the twin.
LMA	6	R		Local Memory Address. The quadword source address in the Shared Memory Pool (see *7.2.10 Shared Memory Pool* on page 198).

The Data Store Address can be any address in the Egress Data Store, but the picocode must ensure that no twin overflow occurs during writing. For example, it is not a good idea to make the LMA point to the last quad-

word in a 64-byte buffer and set NrOfQuadword to four.

Table 63: WREDS Output

		Operand Source		
Name	Size	Direct	Indirect	Description
LMA	6	R		Local Memory Address. The LMA will be the input value of the LMA incremented by the number of quadword transfers completed by the command.

RDEDS (Read Egress Data Store) Command

RDEDS reads from an absolute address in the Egress Data Store. It can be specified if the quadwords are read from DS0 or DS1, or if this decision is made automatically by information in the Dispatch DSU register.

Table 64: RDEDS Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be read: 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
DSControl	2	Imm16(3..2)	Imm12(3..2)	Defines if the quadwords are read from DS0 or DS1: 00 Reads from the default DS 01 Reads from DS0 10 Reads from DS1 11 Reserved
Disp_DSUSel	1	R		Indicates the default DS chosen at Dispatch. 0 DS0 1 DS1
DSA	19	R		Data Store Address. The source address in the Egress Data Store
LMA	6	R		Local Memory Address. The quadword target address in the Shared Memory Pool (see *7.2.10 Shared Memory Pool* on page 198).

When DSControl is set to 00, the quadwords are read from the default DS, which is determined based on the Dispatch DSUSel field.

Table 65: RDEDS Output

		Operand Source		
Name	Size	Direct	Indirect	Description
LMA	6	R		Local Memory Address. The LMA will be the input value of the LMA incremented by the number of quadword transfers completed by the command.

WRIDS (Write Ingress Data Store) Command

WRIDS writes to an absolute address in the Ingress Data Store.

Table 66: WRIDS Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be written: 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
DSA	19	R		Data Store Address. The target address in the Ingress Data Store. The 11 LSBs of the DSA contain the address. The remaining eight MSBs are not used.
LMA	6	R		Local Memory Address. The quadword source address in the Shared Memory Pool (see *7.2.10 Shared Memory Pool* on page 198).

The Data Store Address (DSA) can be any address in the Ingress Data Store but the picocode must ensure that no buffer overflow occurs during writing. For example, it is not a good idea to make the DSA point to the last quadword in a 64-byte buffer and set NrOfQuadword to four.

Table 67: WRIDS Output

		Operand Source		
Name	Size	Direct	Indirect	Description
LMA	6	R		Local Memory Address. The LMA will be the input value of the LMA incremented by the number of quadword transfers completed by the command.

RDIDS (Read Ingress Data Store) Command

RDIDS reads from an absolute address in the Ingress Data Store.

Table 68: RDIDS Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be read: 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
DSA	19	R		Data Store Address. The source address in the Ingress Data Store
LMA	6	R		Local Memory Address. The quadword target address in the Shared Memory Pool (see *7.2.10 Shared Memory Pool* on page 198).

The Data Store Address (DSA) can be any address in the Ingress Data Store. The picocode must ensure that no buffer overflow occurs during reading. For example, it is not a good idea to make the DSA point to the last quadword in a 64-byte buffer and set NrOfQuadword to four.

Table 69: RDIDS Output

		Operand Source		
Name	Size	Direct	Indirect	Description
LMA	6	R		Local Memory Address. The LMA will be the input value of the LMA incremented by the number of quadword transfers completed by the command.

RDMOREI (Read More Quadword From Ingress) Command

After an Ingress frame dispatch, the hardware stores the first N quadwords of the frame in the DataPool. RDMOREI is used to read more quadwords from the Ingress Data Store. It uses two internal registers that are maintained by the Data Store Coprocessor hardware (Current Cell/Twin Address and Next Quadword Address registers) to maintain the current position in the Ingress Data Store and the Shared Memory Pool. During a RDMOREI, the hardware automatically reads the BCB to update the Current/Cell Twin register when a cell-boundary is crossed. RDMOREI can be executed more than once if more quadwords are required.

Table 70: RDMOREI Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be read. 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
CCTA	19	R		Current Cell/Twin Address. The source address in the Ingress Data Store. Initially, this register is set during a dispatch.
NQWA	3	R		Next QuadWord Address.The target address in the DataPool. Initially, this register is set during a dispatch.

Table 71: RDMOREI Output

		Operand Source		
Name	Size	Direct	Indirect	Description
CCTA	19	R		Current Cell/Twin Address. This register is updated by hardware. Executing RDMOREE again reads the next quadwords from Egress Data Store.
NQWA	3	R		Next QuadWord Address. This register is updated by hardware. Executing RDMOREE again causes the quadwords being read to be stored in the next locations in the DataPool.

RDMOREE (Read More Quadword From Egress) Command

After an Egress frame dispatch, the hardware stores the first N quadwords of the frame in the DataPool. RDMOREE is used to read more quadwords from the Egress Data Store. It uses three internal registers that are maintained by the Data Store Coprocessor hardware (Dispatch DSUSel, Current Cell/Twin Address, and Next Quadword Address registers) to maintain the current position in the Egress Data Store and the Shared Memory Pool. During a RDMOREE, the hardware automatically reads the BCB memory to update the Current/Cell Twin register when a twin-boundary is crossed. The RDMOREE can be executed more than once if more quadwords are required.

Table 72: RDMOREE Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NrOfQuadWord	2	Imm16(1..0)	GPR(1..0)	Defines the number of quadwords to be read. 01 1 quadword 10 2 quadwords 11 3 quadwords 00 4 quadwords
DSControl	2	Imm16(3..2)	Imm12(3..2)	Defines if the quadwords are read from DS0 or DS1: 00 Reads from the default DS 01 Reads from DS0 10 Reads from DS1 11 Reserved
Disp_DSUSel	1	R		Indicates the default DS chosen at Dispatch. 0 DS0 1 DS1
CCTA	19	R		Current Cell/Twin Address. The source address in the Egress Data Store. Initially, this register is set during a dispatch.
NQWA	3	R		Next QuadWord Address. The target address in the DataPool. Initially, this register is set during a dispatch.

Table 73: RDMOREE Output

		Operand Source		
Name	Size	Direct	Indirect	Description
CCTA	19	R		Current Cell/Twin Address. This register is updated by hardware. Executing RDMOREE again reads the next quadwords from Egress Data Store.
NQWA	3	R		Next QuadWord Address. This register is updated by hardware. Executing RDMOREE again causes the quadwords being read to be stored in the next locations in the DataPool.

EDIRTY (Update Egress Dirty Quadwords) Command

EDIRTY writes a quadword from the DataPool array to the Egress Data Store if the quadword has been modified since being loaded into the DataPool by a dispatch or a RDMOREE command. The Data Store Coprocessor maintains a register (Dirty Quadword) which indicates when a quadword within the DataPool has been modified. The EDIRTY command uses the Dispatch DSU register to determine which egress Data Store (DS0, DS1, or both) must be updated. When the Current Cell/Twin Address is modified due to a RDMOREE command, all dirty bits are cleared, indicating no quadwords need to be written back to the egress Data Store.

EDIRTY Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
CCTA	19	R		Current Cell/Twin Address. The source address in the Egress Data Store. Initially, this register is set during a dispatch.
NQWA	3	R		Next Quadword Address. Indicates in which cell in the DataPool the Current Cell/Twin is valid. 1,2,3,4 : indicates the first cell (first 4 quadwords) is represented by the Current Cell/Twin Address. 5,6,7,0 : indicates the second cell (second 4 quadwords) is represented by the Current Cell/Twin Address.
DirtyQW	8	R		This register indicates which quadwords need to be updated.
DSControl	2	Imm16(3..2)	Imm12(3..2)	Defines if the quadwords are written to DS0, DS1 or both: 00 Writes to the default DSU. 01 Writes to DS0 10 Writes to DS1 11 Writes to DS0 and DS1
Disp_DSU	2	R		The default DSU, initialized at Dispatch.

Table 74: EDIRTY Output

		Operand Source		
Name	Size	Direct	Indirect	Description
DirtyQW	8	R		Dirty Quadwords. Bits which represent the data quadwords that were updated on this command will be reset to ‘0’.

IDIRTY (Update Ingress Dirty Quadwords) Command

IDIRTY writes a quadword from the DataPool array to the Ingress Data Store if the quadword has been modified since being loaded into the DataPool by a dispatch or a RDMOREI command. The Data Store Coprocessor maintains a register (Dirty Quadword) which indicates when a quadword within the DataPool has been modified. The IDIRTY command uses the Next Quadword Address to determine which cell within the DataPool is represented by the Current Cell/Twin Address. When the Current Cell/Twin Address is modified due to a RDMOREI command, all dirty bits are cleared, indicating no quadwords need to be written back to the Ingress Data Store.

Table 75: IDIRTY Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
CCTA	19	R		Current Cell/Twin Address. The source address in the Ingress Data Store. Initially, this register is set during a dispatch.
NQWA	3	R		Next Quadword Address. Indicates in which cell in the DataPool the Current Cell/Twin is valid. 1,2,3,4 : indicates the first cell (first 4 quadwords) is represented by the Current Cell/Twin Address. 5,6,7,0 : indicates the second cell (second 4 quadwords) is represented by the Current Cell/Twin Address.

Table 75: IDIRTY Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
DirtyQW	8	R		This register indicates which quadwords need to be updated.

Table 76: IDIRTY Output

		Operand Source		
Name	Size	Direct	Indirect	Description
DirtyQW	8	R		Dirty Quadwords. Bits which represent the data quadwords that were updated on this command will be reset to '0'.

LEASETWIN Command

This command leases a 19-bit twin address from the Egress pool of free twins.

Table 77: LEASETWIN Output

		Operand Source		
Name	Size	Direct	Indirect	Description
CCTA	19	R		Current Cell/Twin Address. Contains the address of the newly leased twin.

7.2.4 The Control Access Bus (CAB) Coprocessor

The CAB coprocessor provides interfaces to the CAB arbiter and the CAB for a thread. A thread must load the operands for a CAB access, such as CAB address and data. The protocol to access the CAB is then handled by the CAB interface coprocessor.

7.2.4.1 CAB Coprocessor Address Map

The CAB coprocessor has three scalar registers that are accessible to a thread and are shown in *Table 78*:

Table 78: CAB Coprocessor Address Map

Symbolic Register Name	Register Number	Size (bits)	Access	Description
CABStatus	x'00'	3	R	Status Register Bit 2 Busy Bit 1 0 = write access 1 = read access Bit 0 Arbitration granted
CABData	x'01'	32	R/W	Data to be written to CAB, or Data read from CAB
CABAddress	x'02'	32	W	Address used during last CAB access

7.2.4.2 CAB Access to NP4GS3 Structures

The Control Address Bus (CAB) is the NP4GS3's facility for accessing internal registers. The CAB is assessable via picocode and is used for both configuration and operational functions. CAB addresses consist of three fields and are defined as follows:

Table 79: CAB Address Field Definitions

Island ID	Structure Address	Element Address	Word Addr
5	23		4
32			

The first field, which is comprised of the five most significant bits of the address, selects one of 32 possible functional islands within the device. The correspondence between the encoded functional island value and the functional island name is shown in the *CAB Address, Functional Island Encoding* table below. Although some functional islands have Island_ID values, they are not accessed via the CAB. These functional islands are the Ingress Data Store, Egress Data Store, and Control Store. Structures in these functional islands are accessed via the Data Store Coprocessor and the TSE.

Table 80: CAB Address, Functional Island Encoding

Island_ID	Functional Island Name	Notes
'00000'	Ingress Data Store	1
'00001'	Ingress PMM	
'00010'	Ingress EDS	
'00011'	Ingress SDM	
'00100'	Embedded Processor Complex	
'00101'	SPM	
'00110'	Ingress Flow Control	
'00111'	Embedded PowerPC	
'01000'	Control Store	1
'01111'	Reserved	
'10000'	Egress Data Store	1
'10001'	Egress PMM	
'10010'	Egress EDS	
'10011'	Egress SDM	
'10100'	Configuration Registers	
'10101'	DASL	
'10110'	Egress Flow Control	
'10111-11111'	Reserved	

1. These functional islands are not accessible via the CAB

The second portion of the CAB address consists of the next most significant 23 bits. This address field is segmented into structure address and element address. The number of bits used for each segment can vary from functional island to functional island. Some functional islands contain only a few large structures while others contain many small structures.The structure address addresses an array within the functional island

while the element address addresses an element within the array. The data width of an element is variable and can exceed the 32-bit data width of the CAB.

The third portion of the CAB address consists of a 4-bit word address for selecting 32-bit segments of the element addressed. This address is necessary for moving structure elements wider than 32-bits across the CAB.

7.2.4.3 CAB Coprocessor Commands

The CAB Coprocessor provides the following commands:

CABARB	Arbitrate for CAB Access. Used by a thread to gain access to the CAB. Once access is granted, that thread maintains control of the CAB until it releases the CAB.
CABACCESS	Read/Write CAB. Moves data onto or from the CAB and the attached CAB accessible registers. The source and destination within the DPPU are GPRs
CABPREEMPT	Preempt CAB. Used only by the GFH thread, it allows the GFH to gain control of the CAB for a single read/write access, even if the CAB has already been granted to another thread.

Table 81: CAB Coprocessor Commands Summary

Opcode	Command	Detail Section
0	CABARB	*CABARB (CAB Arbitration) Command* on page 175
1	CABACCESS	*CABACCESS Command* on page 176
3	CABPREEMPT	*CABPREEMPT Command* on page 176

CABARB (CAB Arbitration) Command

CABARB Requests to become a master on the CAB interface or requests to release the CAB after master status has been granted. CABARB does not cause a stall in the CLP even if run synchronously. The CAB coprocessor always indicates that CABARB was executed immediately, even if the arbiter did not grant the CAB interface to the coprocessor. The picocode must release ownership of the CAB interface when it is finished accessing the CAB or a lockout condition could occur for all non-Preempt accesses.

Table 82: CABARB Input

		Operand Source		
Name	Size	Direct	Indirect	Description
Start_NEnd	1	Imm16(0)		1 Start arbitration. 0 Releases arbitration.

CABACCESS Command

Performs a read or write access on the CAB. Before a CAB access can be performed, a CABARB command must have been issued to acquire ownership of the CAB interface.

Table 83: CABACCESS Input

		Operand Source		
Name	Size	Direct	Indirect	Description
Read_NWrite	1		Imm12(0)	1 Performs a CAB read 0 Performs a CAB write
Address	32		GPR(31..0)	The CAB address

Table 84: CABACCESS Output

		Operand Source		
Name	Size	Direct	Indirect	Description
CABData	32	R		Set for CAB read command

CABPREEMPT Command

CABPREEMPT has the same input and output parameters as CABACCESS, except that a high-priority access to the CAB is performed. No CABARB command is required before CABPREEMPT. If any other coprocessor is CAB bus master (because it previously executed a CABARB), CABPREEMPT takes control of the CAB bus and executes the CAB read or write. After command execution, control of the CAB bus returns to the previous owner.

Use CABPREEMPT with care. For example, it might be used in debug mode when the GFH is single stepping one or more other coprocessors. To give a single step command, a CAB write must be executed using the CABREEMPT command because the coprocessor being single stepped may be executing a CABACCESS command and become CAB bus master. If the GFH used the CABACCESS command instead of CABPREEMPT, a deadlock would occur.

7.2.5 Enqueue Coprocessor

The Enqueue Coprocessor manages the interface between a thread and the Completion Unit and manages the use of the FCBPage that is maintained in the Shared Memory Pool. Each thread has three FCBPage locations in which enqueue information about a frame may be maintained. Two of the pages improve the performance of the Completion Unit interface when they are alternated during consecutive enqueues. The picocode written for the thread does not differentiate between these two pages because hardware manages the swap. The thread uses the third page to allow the picocode to create new frames.

When a thread issues an enqueue command, the first FCBPage is marked as in-use. If the other FCBPage is available, the coprocessor is not considered “busy” and will not stall the CLP even if the command was issued synchronously. The Completion Unit fetches the FCBPage from the Shared Memory pool through the Enqueue Coprocessor and provides its information to the EDS (either ingress or egress as indicated by the enqueue command). The FCBPage is then marked as free. If both FCBPages are marked in use, the Enqueue Coprocessor is considered busy and stalls the CLP if a synchronous command initiated enqueue. To guarantee that FCBPage data is not corrupted, enqueue commands must always be synchronous.

Note: When an enqueue command is issued and the other location of the FCBPage becomes the “active”

page, the data is not transferred between the FCBPages and should be considered uninitialized. This is an important consideration for picocode written to handle egress multicast frames.

7.2.5.1 Enqueue Coprocessor Address Map

Table 85: Enqueue Coprocessor Address Map

Name	Register Address	Size	Access	Description
Disp_Label	x'00'	1	R/W	Indicates whether a label was dispatched to the completion unit for this frame. If the nolabel parameter is not passed during an ENQI or ENQE command, then this bit is used to determine if the enqueue will be done with or without a label. 0 Indicates that a label was not passed to the completion unit for this frame. 1 Indicates that a label was passed to the Completion Unit for this frame.
ActiveFCBPage1	x'FC'	384	R/W	Active FCB Page for the current thread. This page is initialized at dispatch.
InActiveFCBPage1	x'FD'	384	R/W	Inactive FCB Page for the current thread. This page is not initialized at dispatch. This array should never be written.
FCBPage2	x'FE'	384	R/W	Alternate FCBPage

FCBPage Format

The FCBPage format varies based on dispatch parameters (ingress or egress frames) and access methods. The FCBPage can be accessed as a defined field, by word, or by quadword. The set of defined fields varies according to the ingress or egress dispatch parameter. Fields are mapped into locations of the Shared Memory Pool. Fields not defined as a multiple of 8 bits are stored in the least significant bits (right justified) of the byte location.

Figure 50: Ingress FCBPage Format

0	1	2	3	4	5	6	7
SP (6)	Abort (1)	GT (1)	FCInfo (4)	WBC (15)		FCBA (12)	
8	9	10	11	12	13	14	15
CurrentBuffer (11)		Not Used (8)	Not Used (8)	TDMU (2)	L3Stk/iDSU (8/4)	PIB (6)	TOS (8)
16	17	18	19	20	21	22	23
TB (16)		iUCnMC (1)	Priority_SF (2)	LID / MID (21 / 17)			
24	25	26	27	28	29	30	31
VLANHdr (16)		Ins_OvlVLAN (2)	FHF (4)	FHE (32)			
32	33	34	35	36	37	38	39
Not Used (8)	Not Used (8)	Not Used (8)	Not Used (8)	Not Used (8)	Not Used (8)	Not Used (8)	Not Used (8)
40	41	42	43	44	45	46	47
CounterControl (14)		CounterData (16)		CounterBlockIndex (20)			

Table 86: Ingress FCBPage Description

Field	FCB Page Offset	Initialized by Dispatch Unit	Enqueue Info	Size (bits)	Description
SP	x'00'	Y	N	6	Source port of frame
Abort	x'01'	Y	N	1	Frame had been marked abort at time of dispatch
GT	x'02'	Y	N	1	Indicates Guided Traffic
FCInfo	x'03'	Y	Y	4	Flow Control color and frame drop information. See *Table 133: Flow Control Information Values* on page 206. Setting this field to x'F' disables flow control. Flow control must be disabled when enqueing to the GDQ, GFQ, GPQ or the discard queue.
WBC	x'04'	Y	N	15	Working byte count. The number of bytes available in the Ingress Data Store for this frame at the time it was dispatched.
FCBA	x'06'	Y	Y	11	Frame Control Block address for the frame dispatched
CurrentBuffer	x'08'	Y	Y	11	Ingress Data Store Buffer Address of the frame dispatched.
TDMU	x'0C'	N	Y	2	Egress Target DMU 00 A 01 B 10 C 11 D
L3Stk/iDSU	x'0D'	N	Y	8/4	L3Stk - When iUCnMC = 0 (frame is multicast), this field contains the value of the DLL termination offset. The DLL termination offset is defined as the number of bytes starting at the beginning of the frame to the position one byte beyond the end of the data link layer. This value is based upon the encapsulation type. Typically, this is the same as the start of the Layer 3 protocol header, an exception would be for MPLS. iDSU - Data Store unit field in the frame header. Indicates where the frame should be stored when entering the egress side. Used when iUCnMC = 1 (frame is unicast).
PIB	x'0E'	N	Y	6	Point in Buffer. Prior to enqueue, indicates location of the first byte of the frame to be sent across the switch interface
TOS	x'0F'	Y	Y	8	IP Type of Service. If the frame is an IP frame, these bits are the TOS field in the IP Header. Otherwise they are initialized to 0's.
TB	x'10'	N	Y	16	Target Blade Vector or Target Blade Address. When in 16-blade mode these 16 bits are used as a target blade vector for multicast frames. In all other modes and for UC frames this is a target blade address. As a target blade vector, bit 15 corresponds to target blade address 0, that is, TB(0:15) In 64 blade mode, valid unicast target blade addresses are 0 through 63, multicast addresses are 512 to 65535.
iUCnMC	x'12'	N	Y	1	Unicast/Multicast indicator 0 Multicast frame 1 Unicast frame
Priority_SF	x'13'	N	Y	2	Priority indicates the user priority assigned to the frame where high priority is indicated by a value of 0. SF is a special field indicator. Bit Description 0 Special Field 1 Priority

Table 86: Ingress FCBPage Description (Continued)

Field	FCB Page Offset	Initialized by Dispatch Unit	Enqueue Info	Size (bits)	Description
LID/MID	x'14'	N	Y	21/17	Lookup ID. Ingress picocode uses to pass information to the egress picocode. Used when iUCnMC = 1 (frame is unicast). Multicast ID: Ingress picocode uses to pass information to the egress picocode. Used when iUCnMC = 0 (frame is multicast).
VLANHdr	x'18'	N	Y	16	VLAN Tag
Ins_OvlVLAN	x'1A'	N	Y	2	Insert or overlay VLAN. Indicates if the VLAN header provided in the VLANHdr scalar register is inserted or overlays an existing VLAN Tag Bit Description 0 Overlay VLAN 1 Insert VLAN
FHF	x'1B'	N	Y	4	Frame Header Format: 4-bit field used by hardware and set up by picocode. Hardware Classifier uses this value on the egress side to determine the starting instruction address for the frame.
FHE	x'1C'	N	Y	32	Frame header extension
CounterControl	x'28'	N	Y	14	Passed to Flow Control for delayed counter manager functions. Bits Description 13 Enable counter operation for this enqueue. Counter updates on an enqueue work only when the target is a target blade. Counter updates do not occur when enqueueing to the GDQ, GFQ, GPQ, or the discard queue. 12 Add/Increment 11:8 Counter Number 7:0 Counter Definition Table Index
CounterData	x'2A'	N	Y	16	Data passed to Ingress Flow Control for delayed counter manager add functions
CounterBlockIndex	x'2C'	N	Y	20	Block Index passed to Ingress Flow Control for delayed counter manager functions

Figure 51: Egress FCBPage Format

0	1	2	3	4	5	6	7
SB (6)	eUCMC (3)	DSUSel (1)	FCInfo (4)	BCI (20)			
8	9	10	11	12	13	14	15
CurrTwin (20)				Type (3)	DSU (2)	QHD (1)	OW (4)
16	17	18	19	20	21	22	23
QID (20)				EtypeAct (3)		EtypeValue (16)	
				DATAFirstTwin (19)			
24	25	26	27	28	29	30	31
SAPtr (10)		DA47_32 (16)		DA31_0 (32)			
32	33	34	35	36	37	38	39
SAInsOvl (2)	MPLS_ VLANDel (2)	CRCAction (2)	DLLStake (6)	TTLAssist (2)	Not Used (8)	Not Used (8)	Not Used (8)
40	41	42	43	44	45	46	47
CounterControl (14)		CounterData (16)		CounterBlockIndex (20)			

Table 87: Egress FCBPage Description

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
SB	'00'	Y	6	Source Blade
eUCMC	'01'	Y	3	Egress Unicast/Multicast bits 000 Unicast 001 First Multicast 010 Middle Multicast 011 Last Multicast 100 Unicast Static Frame enqueue 101 First Multicast Static Frame 110 Middle Multicast Static Frame 111 Last Multicast Static Frame
DSUSel	'02'	Y	1	Dispatch and read more interface uses the indicated DSU
FCInfo	'03'	Y	4	Flow control color information pulled from the frame header by the hardware classifier. See *Table 133: Flow Control Information Values* on page 206
BCI	'04'	Y	20	Byte count indicator. Passed from the queue to the FCBPage, indicates starting byte location within the first twin, number of data buffers, and ending byte location within the last twin.
CurrTwin	'08'	Y	20	At dispatch, indicates the first twin address of the frame

Table 87: Egress FCBPage Description

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
Type	'0C'	Y	3	Type indicates frame type and data store used. Type (2:1) 00 Frame 01 Reserved 10 Reserved 11 Abort Type (0) for GTQ, GPQ 0 DSU0 1 DSU1 Type(0) for GR0, GB0 0 DSU0 1 Both DSU0 and 1 Type(0) GR1, GB1 0 DSU1 1 Both DSU0 and 1
DSU	'0D'	Y	2	Indicates in which DSU(s) the data for this frame is stored. Value is DSU(1:0). Value Description 00 Reserved 01 Stored in DSU 0 10 Stored in DSU 1 11 Stored in both DSU 0 and 1
QHD	'0E'	N	1	Twin Header Qualifier. Indicates type of twin pointed to by CurrTwin. Used for egress frame alteration. 0 Data twin 1 Header twin
OW	'0F'	N	4	Orphan twin weight. Number of twins orphaned by frame alteration actions. When this field is greater than 0, the DATAFirstTwin scalar must be loaded with the location of the first Data Twin. When this field is 0, the EtypeAct and EtypeValue scalar registers can be loaded for insert and overlay Ethertype frame alterations.
QID	'10'	N	20	Flow Queue identifier address
DATAFirstTwin	'14'	N	19	Address of frame's first data twin. Valid when OW is not 0.
EtypeAct	'15'	N	3	Ethertype action. Valid only when OW is set to 0
EtypeValue	'16'	N	16	Ethertype value used in insert / overlay egress frame alteration
SAPtr	'18'	N	10	Source Address Pointer for egress frame alteration. Indicates the SA Array address used by the E-PMM to locate the source address for egress frame alterations. Valid ranges are 0-63.
DA47_32	'1A'	N	16	Most significant bits of a destination address, used by the E-PMM during egress frame alteration
DA31_0	'1C'	N	32	Least significant bits of a destination address, used by the E-PMM during egress frame alteration
SAInsOvl	'20'	N	2	Egress Frame alteration controls for overlay of SA, DA, and Ethertype (when EtypeAct is set to "001") Bit Description 1 Indicates insert (field value 10) 0 Indicates overlay (field value 01) Both bits may not be set to 1. (11 = invalid and 00 = no action)

Table 87: Egress FCBPage Description

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
MPLS_VLANDel	'21'	N	2	Delete VLAN Tag or MPLS Label. Egress frame alteration controls. Remove VLAN Tag (bit 1 set to 1) Remove MPLS Label (bit 0 set to 1)
CRCAction	'22'	N	2	Egress frame alteration controls for modifying the CRC of an Ethernet frame. Value Description 00 No operation 01 Reserved 10 Append CRC 11 Overlay CRC
DLLStake	'23'	N	6	The value of the DLL termination offset. The DLL termination offset is defined as the number of bytes starting at the beginning of the frame to the position one byte beyond the end of the data link layer. This value is based upon the encapsulation type. Typically, this is the same as the start of the Layer 3 protocol header, an exception would be for MPLS.
TTLAssist	'24'	N	2	Egress frame alteration controls for modifying the time to live field in IP headers or the next hop field in IPx headers. Value is TTLAssist(1:0) below: Value Description 00 Disabled 01 IPv4, decrement TTL 10 IPx, increment hop count 11 Reserved
CounterControl	'28'	N	14	Passed to Flow Control for delayed counter manager functions Bits Description 13 Enable counter operation for this enqueue. Counter updates on an enqueue work only when the target is a target port. Counter updates do not occur when enqueing to the GRx, GBx, GFQ, GPQ, GTQ or E-GDQ. Counter updates are supported for the Discard Port (PortID=41) and the Wrap Ports (PortID = 40, 42) 12 Add/$\overline{\text{Increment}}$ 11:8 Counter Number 7:0 Block Definition Index
CounterData	'2A'	N	16	Data passed to Egress Flow Control for delayed counter manager add functions.
CounterBlockIndex	'2C'	N	20	Block Index passed to Egress Flow Control for delayed counter manager functions

FCBPage Initialization During Dispatch

During a dispatch to a thread, the Hardware Classifier and the Dispatch unit provide information about the frame being dispatched that is used to initialize some of the fields within the FCBPage. Values initialized at the time of dispatch are indicated in *Table 86* and *Table 87*. Values that are not indicated as initialized at dispatch are initialized to '0' for the "active" FCBPage.

7.2.5.2 Enqueue Coprocessor Commands

The following commands are supported by the Enqueue Coprocessor:

ENQE	Enqueue Egress. Enqueues to the Egress EDS via the Completion Unit
ENQI	Enqueue Ingress. Enqueues to the Ingress EDS via the Completion Unit
ENQCLR	Enqueue Clear. Clears (sets all fields to zero in) the specified FCBPage

Table 88: Enqueue Coprocessor Commands Summary

Opcode	Command	Detail Section
0	ENQE	*ENQE (Enqueue Egress) Command* on page 183
1	ENQI	*ENQI (Enqueue Ingress) Command* on page 186
2	ENQCLR	*ENQCLR (Enqueue Clear) Command* on page 188

ENQE (Enqueue Egress) Command

ENQE enqueues frames to the Egress target queues.

Table 89: ENQE Target Queues ENQDE command enqueues a frame in one of these Egress target queues

Target Queue	Description
Target Port/Flow Queues	If the Scheduler is disabled, enqueued frames are transmitted on target ports 0 – 39, and logical ports 40-42. If the Scheduler is enabled, enqueued frames are enqueued into the flow queues.
GR0	Enqueued frames are destined for any GDH (or the GFH when it is enabled for data frame processing). A unicast frame must be stored in DS0 when queue GR0 is used. A multicast frame must always be stored in both DS0 and DS1.
GR1	Enqueued frames are destined for any GDH (or the GFH when it is enabled for data frame processing). A unicast frame must be stored in DS1 when queue GR1 is used. A multicast frame must always be stored in both DS0 and DS1.
GB0	Same as GR0, but treated by the Dispatch Unit as lower priority
GB1	Same as GR1, but treated by the Dispatch Unit as lower priority
GFQ	Enqueued frames in this queue are destined for the GFH
GTQ	Enqueued frames in this queue are destined for the GTH
GPQ	Enqueued frames in the queue are destined for the embedded PowerPC
Discard Queue (E-GDQ)	Enqueued frames in this queue are discarded, which involves freeing E-DS space (twins). Frames are only discarded when the MCCA (Multicast Counter Address) enqueue parameter equals zero, or when the Multicast Counter itself has the value 1.

The QID field in the FCBPage selects the Egress target queue. *Table 90* shows the coding of the QID for this selection. ENQE takes a QueueClass as a parameter, and some values of QueueClass automatically set some bits in the QID field to a predefined value.

Table 90: Egress Target Queue Selection Coding

Scheduler Enabled?	QID(19-18)	Target Queue Class	Queue Address	Priority	Target Queue
Yes	00	Flow Queue	QID(10..0)	-	Flow Queue

Table 90: Egress Target Queue Selection Coding

Scheduler Enabled?	QID(19-18)	Target Queue Class	Queue Address	Priority	Target Queue
No	00	Target Port Queue	QID(5..0) (TPQID)	QID(6) (TPPri)	0-39 Ports 40 Wrap to Ingress GFQ 41 Discard (DPQ) 42 Wrap to Ingress GDQ
-	11	GQueue	QID(2..0) (GQID)	-	000 GR0 001 GR1 010 GB0 011 GB1 100 GFQ 101 GTQ 110 GPQ 111 Discard (E-GDQ)

When a frame is enqueued, the parameters from the FCBPage parameters are extracted and passed to the Egress Target Queue.

Table 91: Egress Target Queue Parameters

Queue Type	Queue Select	Enqueue Parameters
Target Port/Flow Queue	QID	QID, BCI, QHD, OW, DATAFirstTwin, SB, CurrTwin, MCCA, MPLS_VLANDel, SAInsOvl, CRCAction, L3Stake, TTLAssist, DSU, SAPtr, DA, FCInfo, CounterControl, CounterData, CounterBlockIndex
GR0, GR1, GB0, GB1, GFQ, GTQ, GPQ	QID	CurrTwin, Type, BCI, MCCA, CounterControl, CounterData, CounterBlockIndex
Discard (E-GDQ)	QID	CurrTwin, Type (see *Table 92*), BCI, MCCA

Table 92: Type Field for Discard Queue

Type	Definition
001	Discard DS0
010	Discard DS1
Others	Reserved

ENQE takes three parameters: the QueueClass, a NoLabel flag, and the FCBPage. According to the Queue Class parameter, bits 19, 18, and 5..0 in the QID field of the FCBPage are changed by the coprocessor according to *Table 94*. Like ENQI, ENQE does not modify the FCBPage.

Table 93: ENQE Command Input

		Operand Source		
Name	Size	Direct	Indirect	Description
QueueClass	5	Imm16(4..0)	GPR(4..0)	Egress Queue Class as defined in *Table 94*.

Table 93: ENQE Command Input

		Operand Source		
Name	Size	Direct	Indirect	Description
NoLabel	1	Imm16(5)	Imm12(5)	0 The CU will use a label if one was dispatched with this frame. This is determined by the status of the Disp_Label register. 1 The Completion Unit will not use a Label for this enqueue. This enqueue is directly executed and is not part of the frame sequence maintenance.
FCBPageID	2	Imm16(7..6)	Imm12(7..6)	00 Active FCBPage is enqueued. 10 FCBPage 2 is enqueued.

Table 94: Egress Queue Class Definitions

QueueClass	QID19	QID18	QID2	QID1	QID0	Target Queue	Notes
0	-	-	-	-	-	Reserved	
1	0	0	-	-	-	Port/Flow queue	
2	0	0	QID(5..0) = 40			Wrap GFQ queue	4
3	0	0	QID(5..0) = 41			Discard queue (DPQ)	
4	0	0	QID(5..0) = 42			Wrap Frame queue	
5	1	1	0	0	S	GRx queue	1
6	1	1	0	1	S	GBx queue	1
7	1	1	1	0	0	GFQ	
8	1	1	1	0	1	GTQ	
9	1	1	1	1	0	GPQ	
10	1	1	1	1	1	Discard queue (GDQ)	3
15	-	-	-	-	-	Any queue	2

1. The ENQE instruction may automatically select the appropriate Data Store or queue, depending on the DSUSel bit.
2. Queue class 15 does not modify any QID bits and allows picocode full control of the target queue.
3. The Type field is modified: bit 2 is set to 0 and the DSU bits are copied to bits 0 and 1 of the Type field.
4. When using Queue Class 2, 3, or 4, and the scheduler is enabled, it is the responsibility of the picocode to insure that QCB 40, 41 and 42 are initialized for the Wrap GRQ (40), the Discard port (41), and wrap data queue (42).

Note: '-' - the field is not modified by the enqueue instruction
S - the value of the DSUSel bit in the FCBPage

ENQI (Enqueue Ingress) Command

ENQI enqueues frames to the Ingress target queues.

Table 95: ENQI Target Queues

Target Queue	Description
Ingress Multicast queue Priorities 0/1	Enqueued frames are treated as multicast frames. Guided frames must be enqueued in a multicast queue, even when their destination is a single port.
Ingress TP queue Priorities 0/1	Enqueued frames are treated as unicast frames. There are a total of 512 Target DMU queues: 256 high priority (priority 0) and 256 low priority.
Ingress Discard queue Priorities 0/1	Enqueued frames are discarded, which involves freeing Ingress buffer space (BCBs and FCBs). Discarding frames on Ingress consumes ingress scheduler slots, i.e. frames are discarded at 58 bytes per slot. The FCBPage's target blade field (TB) must be set to 0.
Ingress GPQ	Enqueued frames are destined for the GPQ. This queue is currently not supported.
Ingress GDQ	Enqueued frames are destined for any GDH (or GFH when enabled for data frame processing)
Ingress GFQ	Enqueued frames are destined for the GFH

Ingress target queues are selected by means of three fields that are part of the FCBPage: iUCnMC, Priority_SF, and TDMU. *Table 96* shows the coding of this selection.

Table 96: Ingress Target Queue Selection Coding

iUCnMC	Priority	SF	TDMU	Target Queue
0	0	0	-	Ingress Multicast queue – priority 0
0	1	0	-	Ingress Multicast queue – priority 1
1	0	0	0 – 3	Ingress TP queue – priority 0
1	1	0	0 – 3	Ingress TP queue – priority 1
1	0	1	0	Ingress Discard queue – priority 0
1	1	1	0	Ingress Discard queue – priority 1
1	x	1	1	Ingress GPQ
1	x	1	2	Ingress GDQ
1	x	1	3	Ingress GFQ

When a frame is enqueued the parameters from the FCBPage are extracted and passed to the Ingress Target Queue. *Table 97* shows the parameters that are passed to the Ingress EDS.

Table 97: Ingress Target Queue FCBPage Parameters

Frame Type	Parameters	Notes
UC Ethernet	FCBA, CounterControl, CounterData, CounterBlockIndex, TB, TDMU, FCInfo, Priority_SF, iUCnMC, LID, iDSU, FHF, FHE, VLANHdr, PIB, Ins_OvlVLAN	1
MC Ethernet	FCBA, CounterControl, CounterData, CounterBlockIndex, TB, Priority_SF, iUCnMC, FCInfo, MID, L3Stk, FHF, FHE, VLANHdr, PIB, Ins_OvlVLAN	1

1. PIB must equal zero (which is the default value after dispatch of an Ingress frame).

ENQI takes three parameters: the QueueClass, a NoLabel flag, and a FCBPage. According to the QueueClass parameter, the Priority_SF and TDMU fields in the FCBPage are modified by the Enqueue Coprocessor according to *Table 99* when passed to the Ingress EDS. For example, to enqueue a unicast frame for

transmission, the picocode prepares the FCBPage, including the Priority and TP fields and invokes ENQI with QueueClass set to 5.

Table 98: ENQI Command Input

		Operand Source		
Name	Size	Direct	Indirect	Description
QueueClass	5	Imm16(4..0)	GPR(4..0)	*Table 99: Ingress-Queue Class Definition* on page 187
NoLabel	1	Imm16(5)	Imm12(5)	0 The CU will use a label if one was dispatched with this frame. This is determined by the status of the Disp_Label register. 1 The Completion Unit will not use a Label for this enqueue. This enqueue is directly executed and is not part of the frame sequence maintenance.
FCBPageID	2	Imm16(7..6)	Imm12(7..6)	00 Active FCBPage 1 is enqueued. 10 FCBPage 2 is enqueued.

Table 99: Ingress-Queue Class Definition

Queue Class	Symbolic Name	iUCnMC	Priority	SF	TDMU	Target Queue
0	DQ	1	-	1	0	Discard queue. Picocode must set the Priority field. TB field must be set to 0 by the picocode. FCInfo field must be set to x'F'.
1	GPQ	1	1	1	1	GPH queue. FCInfo field must be set to x'F'.
2	I-GDQ	1	1	1	2	GDH queue. FCInfo field must be set to x'F'.
4	GFQ	1	1	1	3	GFH queue. FCInfo field must be set to x'F'.
5	TXQ	-	-	0	-	Multicast queue or Target Port queue (transmission queues). Picocode must set iUCnMC and TP fields.
7	ANYQ	-	-	-	-	Any queue. Picocode must set iUCnMC, Priority, SF and TP fields.

Note: A '-' means the FCBPage field is not modified by the ENQI command when passed to the Ingress-EDS.

ENQI does not modify the FCBPage. For example, if the QueueClass parameter is set to TXQ, the SF field is set to '0'. This means that in the SF field received by the Completion Unit and passed to the Ingress-EDS is modified to '0'. The SF field in the FCBPage is not modified.

ENQCLR (Enqueue Clear) Command

ENQCLR takes one parameter, the FCBPage, and fills the entire FCBPage register with zeros.

Table 100: ENQCLR Command Input

		Operand Source		
Name	Size	Direct	Indirect	Description
FCBPageID	2	Imm16(7..6)	Imm12(7..6)	Indicates which FCB Is to be cleared by the command. 00 Active FCBPage is cleared. 10 FCBPage 2 is cleared.

Table 101: ENQCLR Output

		Operand Source		
Name	Size	Direct	Indirect	Description
FCBPage	384	Array		The FCBPage array indicated by the input FCBPageID will be reset to all '0's.

7.2.6 Checksum Coprocessor

The checksum coprocessor generates checksums using the algorithm found in the IETF Network Working Group RFC 1071 Computing the Internet Checksum (available at http://www.ietf.org). As such, it performs its checksum operation on half word data with a half word checksum result.

7.2.6.1 Checksum Coprocessor Address Map

Table 102: Checksum Coprocessor Address Map

Name	Register (Array) Number	Access	Size (Bits)	Description
ChkSum_Stat	x'00'	R	2	Status of Checksum coprocessor Bits Description 1 Insufficient Indicator (if set to 1) 0 Bad Checksum (if set to 1)
ChkSum_Acc	x'01'	R/W	16	When writing this register, the value is a checksum. When reading this register, the value is a Header Checksum (the one's compliment of a checksum).
ChkSum_Stake	x'02'	R/W	10	Pointer into Data Store Coprocessor Arrays where Checksum is to be performed. Bits Description 9:8 Data Store Coprocessor Array Number 7:0 Byte Offset into the Array
ChkSum_Length	x'03'	R/W	8	Working Length remaining in the Checksum calculation.

7.2.6.2 Checksum Coprocessor Commands

Data for the Checksum Coprocessor must be in one of the Data Store Coprocessor's arrays. The commands to the Checksum Coprocessor include:

GENGEN — Generate Checksum. Generates a checksum over a data block with a specified length. Options of this command include initiating a new checksum operation or continuing a checksum where a previous checksum has left off.

Variations of this command include:

- GENIP (Generate IP Checksum)
- GENGENX (Generate Checksum with Cell Header Skip)
- GENIPX (Generate IP Checksum with Cell Header Skip)

CHKGEN — Check Checksum. Checks a checksum over a data block with a specified length. Options of this command include initiating a new checksum operation or continuing a checksum where a previous checksum has left off.

Variations of this command include:

CHKIP (Check IP Checksum)

CHKGENX (Check Checksum with Cell Header Skip)

CHKIPX (Check IP Checksum with Cell Header Skip)

When an IP is indicated, the starting location (i.e. stake) for the Layer 3 header is passed. The hardware determines the length of the IP header from the header length field and loads this value into the Length scalar register. When generating the checksum, a value of zero is substituted for the half word that contains the current checksum.

When Cell Header Skip is indicated, the cell header in the egress frame is skipped in checksum operations. See *The DataPool for Egress Frames* on page 163 for more details of Cell Header Skip.

Table 103: Checksum Coprocessor Commands Summary

Opcode	Command	Detail Section
0	GENGEN	*GENGEN/GENGENX Commands* on page 190
1	GENIP	*GENIP/GENIPX Commands* on page 191
2	CHKGEN	*CHKGEN/CHKGENX Commands* on page 191
3	CHKIP	*CHKIP/CHKIPX Commands* on page 192
4	GENGENX	*GENGEN/GENGENX Commands* on page 190
5	GENIPX	*GENIP/GENIPX Commands* on page 191
6	CHKGENX	*CHKGEN/CHKGENX Commands* on page 191
7	CHKIPX	*CHKIP/CHKIPX Commands* on page 192

GENGEN/GENGENX Commands

Table 104: GENGEN/GENGENX Command Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Load New Stake	1	Imm(15)	Imm(11)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear Accumulation	1	Imm(14)	Imm(10)	ChkSum_Acc contains the seed for the checksum operation.The value in this register indicates whether to use or clear the data in ChkSum_Acc. 0 Use data 1 Clear ChkSum_Acc
Stake Argument	10	Imm(9:0)	GPR(25:16)	The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’.
ChkSum_Stake	10	R		The stake value to be used for the checksum operation if the Load New Stake argument is ‘0’.
ChkSum_Acc	16	R		Contains the seed for the next checksum operation if the “Clear Accumulation” argument is a ‘0’. (Otherwise there is no data in ChkSum_Acc).
ChkSum_Length	8	R		The number of halfwords to read when calculating the checksum.

Table 105: GENGEN/GENGENX/GENIP/GENIPX Command Outputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Return Code	1	CPEI Signal		0 KO, operation failed because there was not enough data in the DataPool 1 OK, operation completed successfully.
ChkSum_Stat	2	R		Status of Checksum Coprocessor Bits Description 1 Insufficient data in DataPool 0 Bad Checksum
ChkSum_Stake	10	R		Stake Register. Points to the halfword of data following the last halfword used in the checksum command.
ChkSum_Acc	16	R		The seed for the next checksum operation Contains the resulting checksum if the Return Code is OK.

GENIP/GENIPX Commands

Table 106: GENIP/GENIPX Command Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Load New Stake	1	Imm(15)	Imm(11)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear Accumulation	1	Imm(14)	Imm(10)	ChkSum_Acc contains the seed for the checksum operation.The value in this register indicates whether or not to use or clear the data in ChkSum_Acc. 0 Use data 1 Clear ChkSum_Acc.
Clear IP Count	1	Imm(13)	Imm(9)	
Stake Argument	10	Imm(9:0)	GPR(25:16)	The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’.
ChkSum_Stake	10	R		The stake value to be used for the checksum operation if the Load New Stake argument is ‘0’.
ChkSum_Acc	16	R		Contains the seed for the next checksum operation if the “Clear Accumulation” argument is a ‘0’. (Otherwise there is no data in ChkSum_Acc).

For GENIP/GENIPX Command Outputs, see *Table 105: GENGEN/GENGENX/GENIP/GENIPX Command Outputs* on page 190.

CHKGEN/CHKGENX Commands

Table 107: CHKGEN/CHKGENX Command Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Load New Stake	1	Imm(15)	Imm(11)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Stake Argument	10	Imm(9:0)	GPR(25:16)	The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’.
ChkSum_Stake	10	R		The stake value to be used for the checksum operation if the Load New Stake argument is ‘0’.
ChkSum_Acc	16	R		This is the checksum to be verified.
ChkSum_Length	8	R		The number of halfwords to read when calculating the checksum.

Table 108: CHKGEN/CHKGENX/CHKIP/CHKIPX Command Outputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Return Code	1	CPEI Signal		0 KO, operation failed. Check status register for reason. 1 OK, operation completed successfully.
ChkSum_Stat	2	R		Status of Checksum Coprocessor Bits Description 1 Insufficient data in DataPool 0 Bad Checksum
ChkSum_Stake	10	R		Stake Register. Points to the halfword of data following the last halfword used in the checksum command.
ChkSum_Acc	16	R		The seed for the next checksum operation. If equal to 0 the checksum was correct.

CHKIP/CHKIPX Commands

Table 109: CHKIP/CHKIPX Command Inputs

		Operand Source		
Name	Size	Direct	Indirect	Description
Load New Stake	1	Imm(15)	Imm(11)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear IP Count	1	Imm(13)	Imm(9)	
Stake Argument	10	Imm(9:0)	GPR(25:16)	The new stake value to be used for the checksum operation if the Load New Stake argument is '1'.
ChkSum_Stake	10	R		The stake value to be used for the checksum operation if the Load New Stake argument is '0'.
ChkSum_Acc	16	R		This is the checksum to be verified.

For CHKIP/CHKIPX Command Outputs, see *Table 108: CHKGEN/CHKGENX/CHKIP/CHKIPX Command Outputs* on page 192.

Results of the commands are found in ChkSum_Acc, ChkSum_Stake, and the 1-bit return code to the CLP. ChkSum_Acc contains the result of the checksum calculation. ChkSum_Stake contains the byte location following the last half word included in the checksum. The return code indicates the operation completed successfully or was verified. The Status register may need to be read to determine the status on a bad return code.

7.2.7 String Copy Coprocessor

The String Copy coprocessor extends the DPPU's capabilities to move blocks of data without tying up the CLP. The data is moved within the Shared Memory Pool and can start and end on any byte boundary within a defined array.

7.2.7.1 String Copy Coprocessor Address Map

Table 110: String Copy Coprocessor Address Map

Name	Register Number	Size (Bits)	Access	Description
StrCpy_SAddr	x‘00’	14	R/W	The source address for the data to be copied: Bit Description 14 Cell header skip access mode 13:10 Coprocessor Number 9:8 Array Number from coprocessor address maps 7:0 Byte Offset within the Array
StrCpy_DAddr	x‘01’	14	R/W	The destination address for the data to be copied: Bit Description 14 Cell header skip access mode 13:10 Coprocessor Number 9:8 Array Number from coprocessor address maps 7:0 Byte Offset within the Array
StrCpy_ByteCnt	x‘02’	8	R	The number of bytes remaining to be moved. This is a working register. Once the coprocessor starts, this register will no longer be valid (it will show the number of bytes remaining).

7.2.7.2 String Copy Coprocessor Commands

Table 111: String Copy Coprocessor Commands Summary

Opcode	Command	Detail Section
0	STRCOPY	*StrCopy (String Copy) Command* on page 193

StrCopy (String Copy) Command

StrCopy moves multiple bytes of data between arrays in the Shared Memory Pool. The CLP passes the starting byte locations of the source and destination data blocks, and the number of bytes to move.

Table 112: StrCopy Command Input

		Operand Source		
Name	Size	Direct	Indirect	Description
StrCpy_SAddr	14	R		Source Address. See *Table 110: String Copy Coprocessor Address Map* on page 193
StrCpy_DAddr	14	R		Destination Address. See *Table 110: String Copy Coprocessor Address Map* on page 193
NumBytes	8	Imm(7:0)	GPR(7:0)	Number of Bytes to transfer

Table 113: StrCopy Command Output

		Operand Source		
Name	Size	Direct	Indirect	Description
StrCpy_SAddr	14	R		Source Address. The offset field of the source address will be incremented by the number of bytes transferred.
StrCpy_DAddr	14	R		Destination Address. The offset field of the destination address will be incremented by the number of bytes transferred.
NumBytes	8	R		This field is 0.

7.2.8 Policy Coprocessor

The Policy Coprocessor provides an interface to the Policy Manager for threads. A thread requests an update to the "color" of a frame through this interface. Frame color is part of the network processor's configurable flow control mechanism which determines what actions may be taken on the frame. A thread must wait until the Policy Manager, via the Policy Coprocessor, returns a result.

7.2.8.1 Policy Coprocessor Address Map

Table 114: Policy Coprocessor Address Map

Name	Register Number	Size (bits)	Access	Description
PolColor	x'00'	2	R/W	Both the color that is passed to the Policy Manager and the result color that is passed back from the Policy Manager
PolPktLen	x'01'	16	R/W	The packet length sent to the Policy Manager
PolCBA	x'02'	20	R	A value of the Policy Control Block Address that was found in a leaf after the frame has been classified and a search was performed

7.2.8.2 Policy Coprocessor Commands

Table 115: Policy Coprocessor Commands Summary

Opcode	Command	Detail Section
0	POLACCESS	*PolAccess (Access Policy Manager) Command* on page 194

PolAccess (Access Policy Manager) Command

PolAccess requests that the Policy Manager accesses the policy control block for the flow that this frame is a member of. Operands include the policy control block address, the length of the packet (usually the IP packet length), and the color currently assigned to the frame. The result returned is a new frame color.

Table 116: PolAccess Input

		Operand Source		
Name	Size	Direct	Indirect	Description
Policy CBA	20		GPR(19:0)	PolCBA Address
PolColor	2	R		Policy Color
PolPktLen	16	R		Policy Packet Length

Table 117: PolAccess Output

		Operand Source		
Name	Size	Direct	Indirect	Description
PolColor	2	R		Returned Policy Color

7.2.9 Counter Coprocessor

The Counter Coprocessor provides an interface to the Counter Manager for threads. The Counter Coprocessor has an eight-deep queue for holding Counter Access commands issued by any of the four threads running in each DPPU. Except for counter reads, the Counter Coprocessor will not stall the CLP on synchronous commands unless the queue is full. This allows for one thread to have multiple counter commands outstanding simultaneously. For example one of the threads may have all the outstanding commands in the queue or each thread may have two each. Normal coprocessor operation would only allow one outstanding command per thread.

7.2.9.1 Counter Coprocessor Address Map

Table 118: Counter Coprocessor Address Map

Name	Register Number	Size (bits)	Access	Description
CtrDataLo	x‘00’	32	R/W	Counter Data Low. This register holds the least significant 32 bits of a counter on a read commands. On write or add commands the lower 16 bits serve as the data passed to the Counter Manager. (Only bits 15..0 are write accessible).
CtrDataHi	x‘01’	32	R	Counter Data High. This register holds the Most significant 32 bits of a counter on a read commands.
CtrControl	x‘02’	12	R/W	Counter Control. The bits have the following meaning: 11:8 = Counter Number - Defines which counter in the CounterBlock should be updated. 7:0 = Block Definition Index - An index in the CounterDefMem.

7.2.9.2 Counter Coprocessor Commands

The Counter Coprocessor provides the following commands:

CtrInc	Counter Increment. Initiates a Modify and Increment command to the Counter Manager.
CtrAdd	Counter Add. Initiates a Modify and Add command to the Counter Manager. The coprocessor passes the Counter Manager a 16-bit value to add to the indicated counter.
CtrRd	Counter Read. Initiates a Read and No Clear command to the Counter Manager. The Counter Manager returns the counter value to the Counter Coprocessor and leaves the counter unmodified.
CtrRdClr	Counter Read with Counter Clear. Initiates a Read and Clear command to the Counter Manager. The Counter Manager returns the counter value to the Counter Coprocessor and resets the counter.
CtrWr15_0	Counter Write Bits 15:0 of a Counter. Initiates a Write Command to the Counter Manager. The coprocessor passes a 16-bit value to be loaded into bits 15:0 of the counter. Uses LSB of counter data low register.
CtrWr31_16	Counter Write Bits 31:16 of a Counter. Initiates a Write Command to the Counter Manager. The coprocessor passes a 16-bit value to be loaded into bits 31:16 of the counter. Uses LSB of counter data low register.

Table 119: Counter Coprocessor Commands Summary

Opcode	Command	Detail Section
0	CTRINC	*CtrInc (Counter Increment) Command* on page 196
1	CTRADD	*CtrAdd (Counter Add) Command* on page 197
4	CTRRD	*CtrRd (Counter Read) / CtrRdClr (Counter Read Clear) Command* on page 197
5	CTRRDCLR	*CtrRd (Counter Read) / CtrRdClr (Counter Read Clear) Command* on page 197
6	CTRWR15_0	*CtrWr15_0 (Counter Write 15:0) / CtrWr31_16 (Counter Write 31:16) Command* on page 197
7	CTRWR31_16	*CtrWr15_0 (Counter Write 15:0) / CtrWr31_16 (Counter Write 31:16) Command* on page 197

CtrInc (Counter Increment) Command

CtrInc performs an access to the central counter manager to increment a counter. This command does not cause synchronous commands to stall if the Counter Coprocessor queue is not full.

Table 120: Ctrinc Input

		Operand Source		
Name	Size	Direct	Indirect	Description
BlockIndex	6 or 20		GPR(19..0)	Defines a CounterBlock within an array of CounterBlocks
CtrControl	12		GPR(11..0)	Counter Control. The bits have the following meaning: 11:8 = Counter Number - Defines which counter in the CounterBlock should be updated. 7:0 = Block Definition Index - An index in the CounterDefMem.

CtrAdd (Counter Add) Command

CtrAdd performs an access to the central counter manager to add a 16-bit value to a counter. This command will not cause synchronous commands to stall if the Counter Coprocessor queue is not full.

Table 121: CtrAdd Input

		Operand Source		
Name	Size	Direct	Indirect	Description
BlockIndex	6 or 20		GPR(19..0)	Defines a CounterBlock within an array of CounterBlocks
CtrDataLo	16		R(15..0)	The value to be added to the counter.
CtrControl	12		GPR(11..0)	Counter Control. The bits have the following meaning: 11:8 = Counter Number - Defines which counter in the CounterBlock should be updated. 7:0 = Block Definition Index - An index in the CounterDefMem.

CtrRd (Counter Read) / CtrRdClr (Counter Read Clear) Command

CtrRd / CtrRdClr performs an access to the central counter manager to read a counter. The CtrRdClr command also clears the counter after the read is performed.

Table 122: CtrRd/CtrRdClr Input

		Operand Source		
Name	Size	Direct	Indirect	Description
BlockIndex	6 or 20		GPR(19..0)	Defines a CounterBlock within an array of CounterBlocks
CtrControl	12		GPR(11..0)	Counter Control. The bits have the following meaning: 11:8 = Counter Number - Defines which counter in the CounterBlock should be updated. 7:0 = Block Definition Index - An index in the CounterDefMem.

Table 123: CtrRd/CtrRdClr Output

		Operand Source		
Name	Size	Direct	Indirect	Description
CtrDataLo	32		R	For 32-bit counters this register contains the value of the counter once the read is performed. For 64-bit counters, this register contains the least significant 32 bits of the counter once the read is performed.
CtrDataHi	32		R	For 32-bit counters this register is not valid. For 64-bit counters, this register contains the most significant 32 bits of the counter once the read is performed.

CtrWr15_0 (Counter Write 15:0) / CtrWr31_16 (Counter Write 31:16) Command

CtrWr15_0/CtrWr31_16 performs an access to the central counter manager to write a 16-bit value to a counter. The CtrWr15_0 writes the 16-bit data value to bits 15..0 of the counter and the CtrWr31_16 writes the value to bits 31..16 of the counter. This command does not cause synchronous commands to stall if the Counter Coprocessor queue is not full.

Table 124: CtrWr15_0/CtrWr31_16 Input

		Operand Source		
Name	Size	Direct	Indirect	Description
BlockIndex	6 or 20		GPR(19..0)	Defines a CounterBlock within an array of CounterBlocks
CtrDataLo	16		R(15..0)	The value to be written to the counter.
CtrControl	12		GPR(11..0)	Counter Control. The bits have the following meaning: 11:8 = Counter Number - Defines which counter in the CounterBlock should be updated. 7:0 = Block D efinition Index - An index in the CounterDefMem.

7.2.10 Shared Memory Pool

The 4 K byte shared memory pool is used by all threads running in the DPPU. 1 K bytes are used by each thread and are subdivided into the following areas per thread:

Table 125: Shared Memory Pool

Quadword Address	Owning Coprocessor	Array
0-2	Enqueue	FCB Page 1A
3	Data Store	Configuration Quadword
4-6	Enqueue	FCB Page 1B
7	CLP	Stack 0
8-10	Enqueue	FCB Page 2
11	CLP	Stack 1
12-15	Data Store	Scratch Memory 0
16-23	Data Store	DataPool
24-31	Data Store	Scratch Memory 1
32-47	TSE	Tree Search Results Area 0
40-47	TSE	Tree Search Results Area 1
48-63	TSE	Tree Search Results Area 2
56-63	TSE	Tree Search Results Area 3

7.3 Interrupts and Timers

The NP4GS3 provides a set of hardware and software interrupts and timers for the management, control, and debug of the processor. When an interrupt event occurs or a timer expires a task is scheduled to be processed by the Embedded Processor Complex. The interrupt or timer task does not preempt threads currently processing other tasks, but is scheduled to be dispatched to the next idle thread that is enabled to process the interrupt or timer. The starting address of the task is determined by the Interrupt or Timer Class and read from *Table 128: Port Configuration Memory Content* on page 201.

7.3.1 Interrupts

The interrupt mechanism within the NP4GS3 has several registers: the Interrupt Vector Registers 0-3, Interrupt Mask Register 0-3, Interrupt Target Register 0-3. and the Software Interrupt Registers.

7.3.1.1 Interrupt Vector Registers

The Interrupt Vector Register is a collection of interrupts that will share a common code entry point upon dispatch to a thread in the EPC. A bit representing the individual interrupt within the Interrupt Vector Register is only set on the initial event of the interrupt. Even if the Interrupt Vector Register is cleared, an outstanding interrupt will not set the bit in the register again until it is detected that the interrupt condition is going from an inactive to active state. Picocode can access the Interrupt Vector registers either through the dashboard or through the master copy. The Interrupt Vector Register is cleared when it is read from its Master Copy address. When read from the dashboard, the register is not cleared.

7.3.1.2 Interrupt Mask Registers

The Interrupt Mask Registers have a bit to correspond with each bit in the Interrupt Vector Registers. The Interrupt Mask Registers indicate which interrupts in the Interrupt Vector Registers cause an task to be scheduled for processing by the EPC. The Interrupt Mask Registers have no affect on the setting of the Interrupt Vector Registers.

7.3.1.3 Interrupt Target Registers

The Interrupt Target Register indicates which Thread types in the EPC are enabled to process the interrupt of a given class 0-3.

7.3.1.4 Software Interrupt Registers

The Software Interrupt Registers provide 12 unique interrupts (three in each of the four classes) that can be defined by Software and accessed through CAB accesses. Writing a Software Interrupt Register sets the corresponding bit within the Interrupt Vector Register 0-3 and has the same effect as a hardware defined interrupt.

7.3.2 Timers

The NP4GS3 has four Timer Interrupt Counters that can be used to generate delayed interrupts to the EPC.

7.3.2.1 Timer Interrupt Counters

Timer Interrupt Counters 0-2 are 24 bits in length and decrement at 1 ms intervals. Timer Interrupt Counter 3

is 32 bits in length and decrements at 10 µs intervals. The Timer Interrupt Counters are activated by writing a non-zero value to the register. When the Timer decrements to a zero value it will schedule a timer task to be processed by the EPC

7.3.3 Port Configuration Memory

Table 128: Port Configuration Memory Content on page 201 provides control, default, and software defined information on each dispatch and is passed to the thread to be stored in the Configuration Quadword array in the Data Store Coprocessor.

7.3.3.1 Port Configuration Memory Index Definition

The Port Configuration Memory Index consists of 64 entries that are indexed based upon multiple parameters including Ingress port, Egress queues, timers, and interrupts as defined in *Table 126*.

Table 126: Port Configuration Memory Index

Port Configuration Memory Index	Definition
0 .. 39	Ingress SP (from GDQ)
40	Ingress Wrap Frame (I-GDQ)
41	Reserved
42	I-GFQ
43	Ingress Wrap Guided
44	Reserved
45	GPQ
46	Egress GFQ
47	GTQ
48	Egress frame in either DS0 or DS1
49	Egress frame in DS0 and DS1
50	Reserved
51	Reserved
52	Reserved
53	Reserved
54	Egress Abort of frame in either DS0 or DS1
55	Egress Abort of frame in DS0 and DS1
56	Interrupt 0
57	Interrupt 1
58	Interrupt 2
59	Interrupt 3
60	Timer 0
61	Timer 1
62	Timer 2
63	Timer 3

Table 127: Relationship Between SP Field, Queue, and Port Configuration Memory Index

SP field in FCB1	Queue	Port Configuration Memory Index
0 .. 39 (denotes physical port)	GDQ	0 .. 39
0 .. 39 (denotes physical port)	I-GFQ	42
40 (denotes wrap port)	GDQ	40
40 (denotes wrap port)	I-GFQ	43

7.3.4 Port Configuration Memory Contents Definition

Bits 127 .. 24 are software defined and are for use by the picocode. The remaining bits are used by the hardware, as defined in *Table 128*.

Table 128: Port Configuration Memory Content

Field Name	Bits	Description
	127 .. 24	For Software use.
POS_AC	23	0 AC not Present 1 AC Present
Ethernet/PPP	22	0 PPP port 1 Ethernet port
CodeEntryPoint	21 .. 6	The default code entry point. Can be overwritten by the hardware classifier (if enabled).
CULabGenEnabled	5	0 No label is generated. 1 The Hardware Classifier generates a label that is used by the Completion Unit to maintain frame sequence. This field must be set to 0 for Port Configuration Memory entries 54, 55 (representing aborted frames on the Egress), 56-59 (interrupts), and 60-63 (timers).
HardwareAssistEnabled	4	0 Hardware Classifier is disabled 1 Hardware Classifier is enabled and the classifier may overwrite the CodeEntryPoint
Reserved	3 .. 2	Reserved. Set to ‘00’.
NumberOfQuadWords	1 .. 0	Defines the number of quadwords that the Dispatch Unit will read from the frame and store in the DataPool. A value of ‘00’ represents four quadwords. For Port Configuration Memory entries 56-63 (interrupts and timers), this field is ignored and the Dispatch Unit will not write any quadword into the DataPool.

7.4 Hardware Classifier

The Hardware Classifier provides hardware assisted parsing of the ingress and egress frame data that is dispatched to a thread. The results are used to precondition the state of a thread by initializing the thread's General Purpose and Coprocessor Scalar Registers along with the registers' resources and a starting instruction address for the CLP. Parsing results indicate the type of Layer 2 encapsulation, as well as some information about the Layer 3 frame. Recognizable Layer 2 encapsulations include PPP, 802.3, DIX V2, LLC, SNAP header, and VLAN tagging. Reported Layer 3 information includes IP and IPX network protocols, five programmable network protocols, the detection of option fields, and IP transport protocols (UDP and TCP). If enabled, the Hardware Classifier also generates labels that are passed to the Completion Unit with a thread identifier. The CU uses them to maintain frame order within a flow.

7.4.1 Ingress Classification

Ingress classification, the parsing of frame data which originated in the Ingress EDS and is now being passed from the Dispatch Unit to the DPPU, can be applied to Ethernet/802.3 frames with the following Layer 2 encapsulation: DIX V2, 802.3 LLC, SNAP header, and VLAN Tagging. Classification can also be done on POS frames using the Point-to-Point Protocol with or without the AC Field Present (POS_AC) field.

7.4.1.1 Ingress Classification Input

The Hardware Classifier needs the following information to classify an ingress frame:

- Port Configuration Memory Table Entry (*Table 128* on page 201). The Hardware Classifier uses the following fields:
  - CodeEntryPoint - the default starting instruction address.
  - HardwareAssistEnabled
  - CULabGenEnabled
  - Ethernet/PPP
  - POS AC
- Frame Data. Four quadwords must be dispatched (per frame) for ingress classification.
- Protocol Identifiers: The Hardware Classifier compares the values in the Protocol Identifier registers with the values of the fields in the frame that correspond to those identifiers to determine if one of the configured protocols is encapsulated in the frame. The Hardware Classifier supports seven Ethernet Protocols (five of which are configurable) and eight Point-to-Point Protocols (five of which are configurable). Two registers need to be configured for each Ethernet Protocol, the Ethernet Type value and an 802.2 Service Access Point value. The Protocol Identifiers are configured from the CAB.

Table 129: Protocol Identifiers

CAB Address	Access	Bits	Description
x‘2500 0000’	R/W	16	Ethernet Type for Protocol 0
x‘2500 0010’	R/W	16	Ethernet Type for Protocol 1
x‘2500 0020’	R/W	16	Ethernet Type for Protocol 2
x‘2500 0030’	R/W	16	Ethernet Type for Protocol 3
x‘2500 0040’	R/W	16	Ethernet Type for Protocol 4
x‘2500 0050’	R	16	Ethernet Type for IPX (x‘8137’)
x‘2500 0060’	R	16	Ethernet Type for IP. (x‘0800’)
x‘2500 0070’		16	Reserved
x‘2500 0080’	R/W	16	Point to Point Type for Protocol 0
x‘2500 0090’	R/W	16	Point to Point Type for Protocol 1
x‘2500 00A0’	R/W	16	Point to Point Type for Protocol 2
x‘2500 00B0’	R/W	16	Point to Point Type for Protocol 3
x‘2500 00C0’	R/W	16	Point to Point Type for Protocol 4
x‘2500 00D0’	R	16	Point to Point Type for IPX (x002B’)
x‘2500 00E0’	R	16	Point to Point Type for IP. (x0021’)
x‘2500 00F0’	R	16	Point to Point Control Type Frame (MSBs of type field is ‘1’)
x‘2500 0100’	R/W	8	Service Access Point for Protocol 0
x‘2500 0110’	R/W	8	Service Access Point Type for Protocol 1
x‘2500 0120’	R/W	8	Service Access Point Type for Protocol 2
x‘2500 0130’	R/W	8	Service Access Point Type for Protocol 3
x‘2500 0140’	R/W	8	Service Access Point Type for Protocol 4
x‘2500 0150’	R	8	Service Access Point Type for IPX (x‘E0’)
x‘2500 0160’	R	8	Service Access Point Type for IP. (x‘06’)
x‘2500 0170’		8	Reserved

7.4.1.2 Ingress Classification Output

The outputs of the ingress hardware classification are:

- The Starting Instruction Address that is stored in the HCCIA table. This address is only used when the hardware classification is enabled. When the hardware classification is disabled, or if a protocol match is not found, the Code Entry Point from the Port Configuration Memory table (*Table 128: Port Configuration Memory Content* on page 201) is used as the Starting Instruction Address and is passed to the thread. If a protocol match does occur, the starting instruction address is retrieved from the last 24 entries of the HCCIA table. HCCIA values are configured from the CAB. The address into HCCIA are provided in *Table 130: HCCIA Table* on page 204.

Table 130: HCCIA Table

HCCIA CAB Address	Bits	Classification
x‘2500 0400’ x‘2500 05F0’	16	Egress Locations (see section *7.4.2.1 Egress Classification Input* on page 206)
x‘2500 0600’	16	Ethernet Protocol 0 classification with no 802.1Q VLAN
x‘2500 0610’	16	Ethernet Protocol 1 classification with no 802.1Q VLAN
x‘2500 0620’	16	Ethernet Protocol 2 classification with no 802.1Q VLAN
x‘2500 0630’	16	Ethernet Protocol 3 classification with no 802.1Q VLAN
x‘2500 0640’	16	Ethernet Protocol 4 classification with no 802.1Q VLAN
x‘2500 0650’	16	Ethernet IPX classification with no 802.1Q VLAN
x‘2500 0660’	16	Ethernet IP classification with no 802.1Q VLAN
x‘2500 0670’	16	Ingress Aborted Frame at time of dispatch
x‘2500 0680’	16	Ethernet Protocol 0 classification with 802.1Q VLAN
x‘2500 0690’	16	Ethernet Protocol 1 classification with 802.1Q VLAN
x‘2500 06A0’	16	Ethernet Protocol 2 classification with 802.1Q VLAN
x‘2500 06B0’	16	Ethernet Protocol 3 classification with 802.1Q VLAN
x‘2500 06C0’	16	Ethernet Protocol 4 classification with 802.1Q VLAN
x‘2500 06D0’	16	Ethernet IPX classification with 802.1Q VLAN
x‘2500 06E0’	16	Ethernet IP classification with 802.1Q VLAN
x‘2500 06F0’	16	Ethernet VLAN frame with an ERIF
x‘2500 0700’	16	Point to Point Protocol 0 classification
x‘2500 0710’	16	Point to Point Protocol 1 classification
x‘2500 0720’	16	Point to Point Protocol 2 classification
x‘2500 0730’	16	Point to Point Protocol 3 classification
x‘2500 0740’	16	Point to Point Protocol 4 classification
x‘2500 0750’	16	Point to Point IPX classification
x‘2500 0760’	16	Point to Point IP classification
x‘2500 0770’	16	Point to Point Control Frame

- Ingress Protocol Type Register

  Contains the output of the Ingress Hardware Classifier. This Data Store Coprocessor register is loaded with the protocol identifier that identifies the frame for the given encapsulation type.

Table 131: Protocol Identifiers for Frame Encapsulation Types

Frame Encapsulation Type	Data Store Coprocessor Register Setting
SNAP	Ethernet Type
DIX V2	Ethernet Type
SAP	DSAP/SSAP
Point to Point Protocol	Point to Point Type

The fields in the frame data that correspond to Data Store Coprocessor Register Settings (see *Table 131*) are passed to the Ingress Protocol Type register. If a VLAN tagged frame with an E-RIF field is present within the frame, or if the Hardware Classifier is disabled, this field is invalid.

- DLL termination offset

  If the Hardware Classifier is enabled, the DLL termination offset is loaded into GPR R0. The DLL termination offset is defined as the number of bytes, starting at the beginning of the frame, to the position one byte beyond the end of the data link layer. This value is based upon the encapsulation type. Typically, this is the same as the start of the Layer 3 protocol header, an exception would be for MPLS.

- Classification Flags:

  If the Hardware Classifier is enabled, classification flags will be loaded into the thread's GPR R1.

Table 132: General Purpose Register Bit Definitions for Ingress Classification Flags

Bit	Definition
Bit 15	Protocol 7 detected
Bit 14	Protocol 6 detected
Bit 13	Protocol 5 detected
Bit 12	Protocol 4 detected
Bit 11	Protocol 3 detected
Bit 10	Protocol 2 detected
Bit 9	Protocol 1 detected
Bit 8	Protocol 0 detected
Bit 7	'0'
Bit 6	Indicates IP Options field present
Bit 5	DIX encapsulation
Bit 4	Indicates SAP encapsulation
Bit 3	Indicates SNAP encapsulation
Bit 2	Indicates 802.1q VLAN frame
Bit 1	Indicates the 802.1q VLAN ID was non-zero
Bit 0	Indicates an ERIF present

- Flow Control Information

The Hardware Classifier initializes the FCBPage's Flow Control Information (FCInfo) field. The field's value is based on IP frame information that includes the frame color indicated by bits 4:3 of the IP header's TOS field and the TCP header's SYN bit. The Hardware Classifier never sets FCInfo field to '1111' but once the field is in the FCBPage it can be written by picocode to be a '1111'.

Table 133: Flow Control Information Values

FCInfo	Definition
0000	TCP - Green
0001	TCP - Yellow
0010	TCP - Red
0011	Non-IP
0100	UDP - Green
0101	UDP - Yellow
0110	UDP - Red
0111	Reserved
1000	TCPSYN- Green
1001	TCPSYN - Yellow
1010	TCPSYN - Red
1011	Reserved
1100	Other IP - Green
1101	Other IP - Yellow
1110	Other IP - Red
1111	Disable Flow Control

7.4.2 Egress Classification

Egress classification, the parsing of frame that originated in the Egress EDS and is being transferred from the Dispatcher to the DPPU is limited to choosing a starting instruction address and generating a label to pass to the Completion Unit.

7.4.2.1 Egress Classification Input

For egress frames, the Hardware Classifier needs the following information to classify the frame:

- Port Configuration Memory Table Entry (*Table 128* on page 201). The Hardware Classifier uses the following fields:
  - CodeEntryPoint- the default starting instruction address
  - HardwareAssistEnabled
  - CULabGenEnabled
- Frame Data. Two quadwords must be dispatched (per frame) for egress classification.

7.4.2.2 Egress Classification Output

The outputs of the Egress hardware classification are:

- Starting Instruction Address.

  The starting instruction address stored in the HCCIA memory is only used when the hardware classification is enabled. When the hardware classification is disabled, the Code Entry Point from the Port Configuration Memory table (*Table 128: Port Configuration Memory Content* on page 201) is used as the Starting Instruction Address and is passed to the thread. If a protocol match does occur, the starting instruction address is retrieved from the first 32 entries of the HCCIA. The address into HCCIA is given by the UC field and by the FHF field in the frame header:

Table 134: HCCIA Index Definition

1	4
UC	FHF

- Frame Data Offset

  GPR R0 is always (i.e. also when the Hardware Classification is disabled) set according to *Table 58: Egress Frames DataPool Quadword Addresses* on page 164

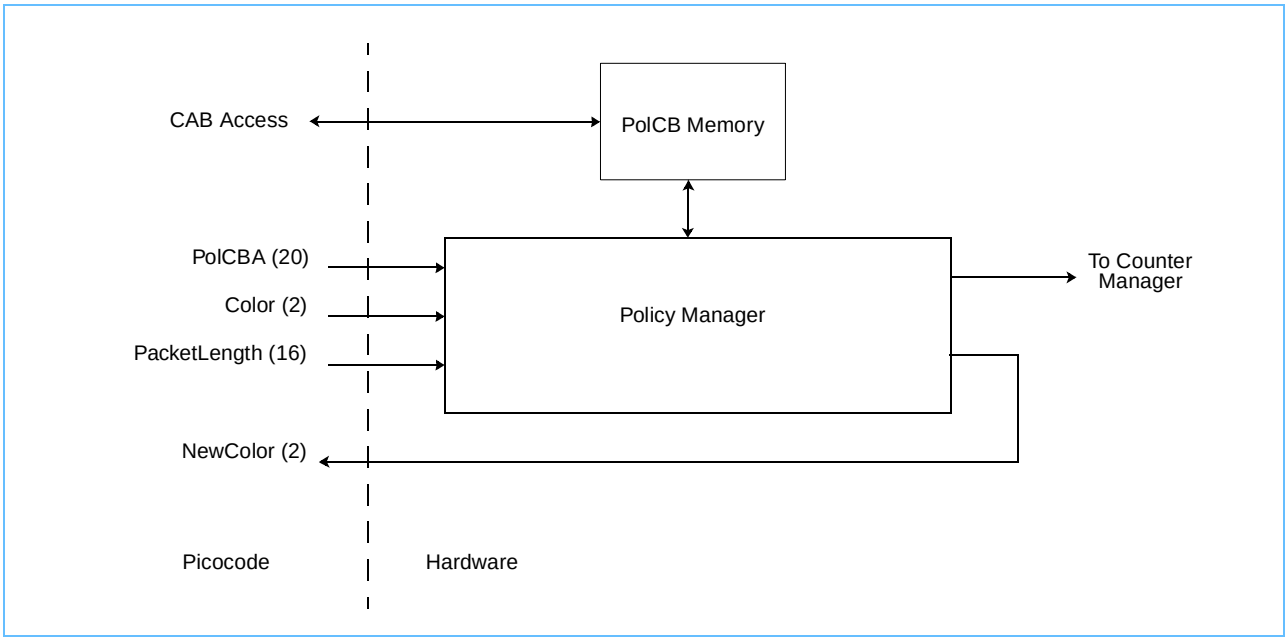
7.5 Policy Manager

The Policy Manager is a hardware assist of the Embedded Processor Complex that performs policy management on up to 1 K ingress flows. It supports four management algorithms. One algorithm pair is "Single Rate Three Color Marker," operated in color-blind or color aware mode. The other is "Two Rate Three Color Marker," operated again in color-blind or color-aware mode. The algorithms are specified in IETF RFCs 2697 and 2698. The algorithms are specified in IETF RFCs 2697 and 2698 (available at http://www.ietf.org).

The Policy Manager maintains up to 1024 leaky bucket meters with selectable parameters and algorithms. The picocode sends the Policy Manager a Policy Management Control Block Address (PolCBA), a Color, and a Packet Length for each incoming packet. According to the Policy Management Control Block (PolCB), two token counters stored in internal memory are regularly incremented (subject to an upper limit) by a rate specified in the PolCB and, when a packet arrives, are decremented by one of four possible algorithms (depending upon the incoming Packet Length). After both actions are complete, the token counters generally have new values and a new color is returned to the picocode.

In addition there are three 10-bit wide packet counters in the PolCB (RedCnt, YellowCnt, and GreenCnt) that use the output packet colors to count the number of bytes (with a resolution of 64 bytes) of each color. When these counters overflow, the Policy Manager invokes the Counter Manager with an increment instruction and counters maintained by the Counter Manager are used for the overflow count. Counter Definition 0 is reserved for the Policy Manager, and must be configured for these overflow counts.

Figure 52: Split between Picocode and Hardware for the Policy Manager



The PolCB must be configured before use. Configuration is accomplished via the CAB. The contents of the PolCB are illustrated in *Table 135: PolCB Field Definitions* on page 209.

Classification of a frame by the picocode must result in a PolCBA. The Hardware Classifier provides the color of the frame in the FCBPage. The frame length used must be parsed from the IP packet header by the picocode.

The Policy Manager receives the following inputs from the picocode:

- PolCBA (20 bits)
- Color (2 bits), the color of the incoming packet. As encoded in the DS Byte, 00 = green, 01 = yellow, 10 = red
- Packet Length (in bytes, 16 bits)

The Policy Manager reads the PolCB from the memory, executes the algorithm configured by the type field in the PolCB, writes the updated PolCB back to the memory and returns the new color to the picocode. Pico-code might use this information as follows

- Perform no special action
- Discard the packet
- Change the DS Byte (i.e. (re-)mark the frame)

Table 135: PolCB Field Definitions

Field	Size	Description
Type	4	Algorithm type. This field must be initialized by picocode. 0000 Color blind single rate three color marker 0001 Color aware single rate three color marker 0010 Color blind two rate three color marker 0011 Color aware two rate three color marker 0100-1111 Reserved
PA_Time	32	Previous arrival time in ticks. This field must be initialized to 0 by the picocode. PA_Time is compared to a running 32 bit counter which is incremented every 165/150ns. Selection of the accuracy of the counter tick is controlled by the setting of the DRAM Parameter Register bit 22 (11/$\overline{10}$). When set to 1 the tick rate is 165ns when set to 0 the tick rate is 150 ns.
C_Token	26	Token Counter for Committed rate accounting in bytes. This field must be initialized by picocode to contain the same value as C_BurstSize. (Format is 17.9)
EP_Token	26	Token Counter for Excess or Peak rate accounting in bytes. This field must be initialized by picocode to contain the same value as EP_BurstSize. (Format is 17.9)
CIR	12	Committed Information Rate in bytes/tick used for two rate algorithms. CIR uses an exponential notation X * 8^{Y-3} where 11:3 X 2:0 Y; valid values of Y are ‘000’ through ‘011’ A tick for the policy manager is defined to be either 165 or 150 ns. Selection of the value for a tick is controlled by the setting of the DRAM Parameter Register bit 22 (11/$\overline{10}$). When set to 1 the tick is 165ns, when set to 0 the tick is 150ns. This field must be initialized by picocode to meet the service level agreement. The PIR must be equal to or greater than the CIR. CIR can be defined in the range of 100Kb/s through 3Gb/s.

Table 135: PolCB Field Definitions (Continued)

Field	Size	Description
PIR	12	Peak Information Rate in bytes/tick used for two rate algorithms. PIR uses an exponential notation X * 8^{Y-3} where 11:3 X 2:0 Y; valid values of Y are '000' through '011' A tick for the policy manager is defined to be either 165 or 150 ns. Selection of the value for a tick is controlled by the setting of the DRAM Parameter Register bit 22 (11/10). When set to 1 the tick is 165ns, when set to 0 the tick is 150ns. This field must be initialized by picocode to meet the service level agreement. The PIR must be equal to or greater than the CIR. PIR can be defined in the range of 100Kb/s through 3Gb/s.
C_Burstsize	17	Committed burst size in bytes. This field must be initialized by picocode to meet the service level agreement. For the single rate algorithms, either the C_BurstSize or the EP_BurstSize must be larger than 0. It is recommended that when the value of C_BurstSize or the EP_BurstSize is larger than 0, it is larger than or equal to the size of the MTU for that stream. For the two rate algorithms, C_BurstSize must be greater than 0. It is recommended that it is larger than or equal to the size of the MTU for that stream.
EP_Burstsize	17	Excess or Peak Burst Size in bytes. Definition depends on algorithm selected. This field must be initialized by picocode to meet the service level agreement. For the single rate algorithms, either the C_BurstSize or the EP_BurstSize must be larger than 0. It is recommended that when the value of C_BurstSize or the EP_BurstSize is larger than 0, it is larger than or equal to the size of the MTU for that stream. For the two rate algorithms, EP_BurstSize must be greater than 0. It is recommended that it is larger than or equal to the size of the MTU for that stream.
GreenCnt	10	Number of bytes (with 64 byte resolution) in packets flagged as "green" by the Policy Manager. When this counter overflows, the Policy Manager uses the Counter Manager interface to increment an extended range counter. This field must be initialized to 0 by picocode. The counter control block for the Policy Manager must be configured at Counter Definition Table entry 0. The Green Count is counter number 0.
YellowCnt	10	Number of bytes (with 64 byte resolution) in packets flagged as "yellow" by the Policy Manager. When this counter overflows, the Policy Manager uses the Counter Manager interface to increment an extended range counter. This field must be initialized to 0 by picocode. The counter control block for the Policy Manager must be configured at Counter Definition Table entry 0. The Yellow Count is counter number 1.
RedCnt	10	Number of bytes (with 64 byte resolution) in packets flagged as "red" by the Policy Manager. When this counter overflows, the Policy Manager uses the Counter Manager interface to increment an extended range counter. This field must be initialized to 0 by picocode. The counter control block for the Policy Manager must be configured at Counter Definition Table entry 0. The Red Count is counter number 2.

7.6 Counter Manager

The Counter Manager is a hardware assist engine used by the EPC to control various counts used by the picocode for statistics, flow control, and policy management. The Counter Manager is responsible for counter updates, reads, clears, and writes, and it allows the picocode to access these functions using single instructions. The Counter Manager arbitrates between all requestors and acknowledges when the requested operation is completed. The Counter Manager works in concert with the Counter Coprocessor logic in order to allow the picocode access to the various counters.

The Counter Manager supports the following:

- 64-bit Counters
- 32-bit Counters
- 24/40-bit Counters
- Read, Read/Clear, Write, Increment, and Add functions
- A maximum of 1 K 64-bit, 2 K 32-bit, or some mix of the two not to exceed a total size of 64 K bits, of fast internal counters
- Up to 4 M 64-bit or 32-bit external counters. Two 32-bit counters are packed into a 64-bit DRAM line. Selection of the counter is accomplished by the low-order address bit.
- Counter Definition Table for defining counter groups and storage locations
- Interfaces to all eight DPPUs, the Policy Manager, Ingress and Egress Flow Control
- Five Independent Counter Storage locations

Figure 53: Counter Manager Block Diagram

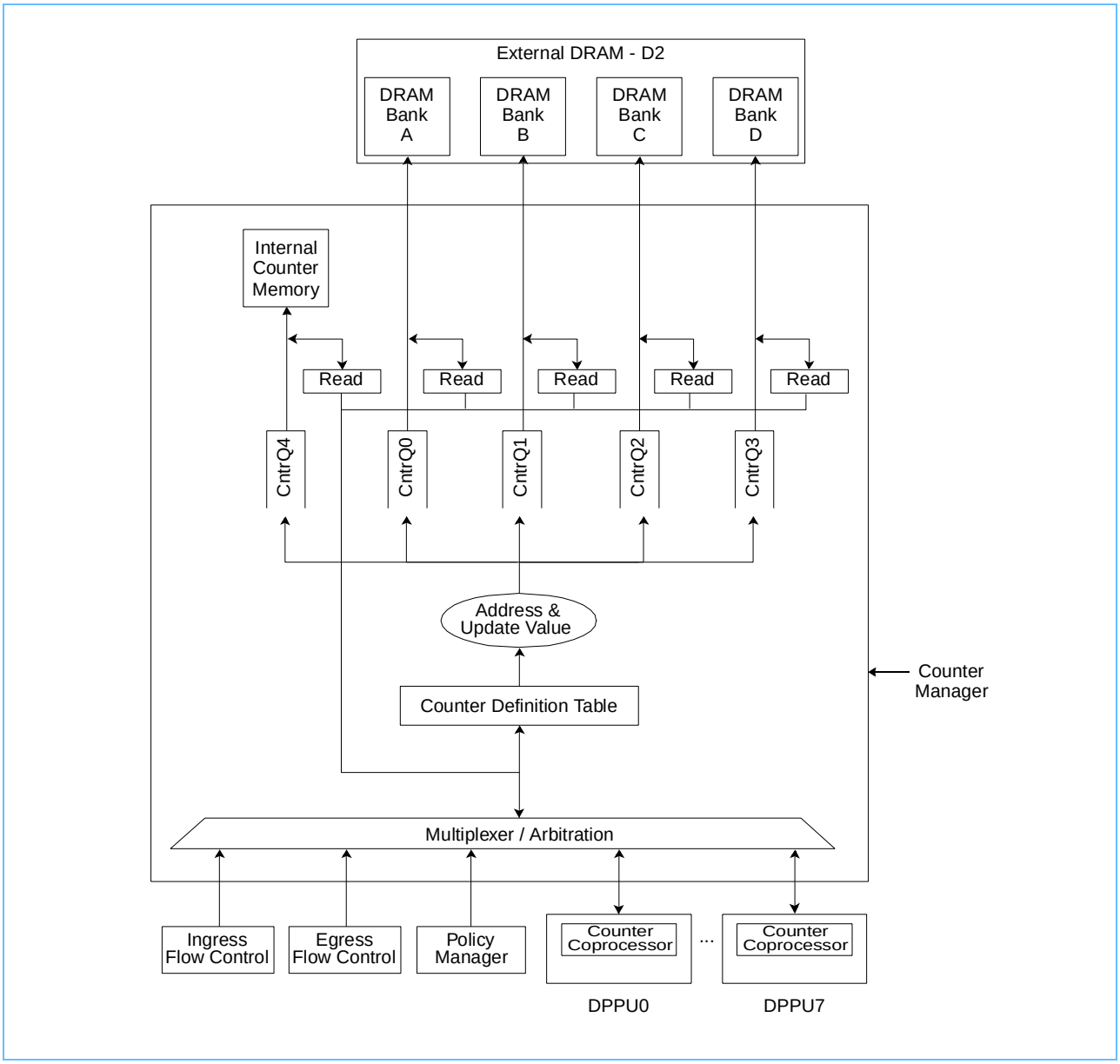


Table 136: Counter Manager Components

Component Name	Description
Counter Definition Table	Contains definitions for each counter block (256 entries). A counter block is made up of the counter's memory storage location (Bank A, Bank B, Bank C, Bank D, or Internal), the base address within that memory, the number of counters within the set, and the counter size.
Multiplexing and Arbitration	Multiplexing and arbitration logic selects the next counter action from all requestors according to priority. Flow Control has the highest priority, Policy Manager the next, and the set of DPPUs the lowest priority. Multiplexing and arbitration logic uses two work conserving round-robins: one between the two Flow Control requestors and one for the set of DPPUs. This logic returns the read data to the appropriate requestor during counter reads.
Address and Update Value	Address and Update Value logic uses information gathered from the Counter Definition Table and the parameters passed from the requestor to create the final counter address and update value.
CntrQ0 - CntrQ4	Five Counter Queues used to temporarily hold the counter request and allow the Counter Manager to access all five memory locations independently. The request is placed into the appropriate queue based on the memory storage location information found in the Counter Definition Table.
Read	Read logic gathers the read data from the appropriate memory location and returns it to the requestor.
Internal Counter Memory	Internal Counter Memory holds the "fast" internal counters and can be configured to hold 64-bit, 24/40-bit, or 32-bit counters. The Internal Counter Memory size is 1024 locations x 64 bits.

Table 137: Counter Types

Type Name	Description
64-bit Counter	Counter that holds up to a 64-bit value.
24/40-bit Counter	Special 64-bit counter that has a standard 40-bit portion and a special 24-bit increment portion. The 40-bit portion is acted upon by the command passed and the 24-bit portion is incremented. This counter allows "byte count" and "frame count" counters to be in one location; the 40-bit portion is the byte count, and the 24-bit portion is the frame count.
32-bit Counter	Counter that holds up to a 32-bit value.

Table 138: Counter Actions

Action Name	Description
Read	Read Counter and return value (either 64 or 32 bits) to the EPC.
Read/Clear	Read Counter and return value (either 64 or 32 bits) to the EPC. Hardware then writes counter to zero.
Write	Write 16 bits to the counter (either bits 31:16 or 15:0) and zero all other bits.
Increment	Add one to the counter value and store updated value in memory.
Add	Add 16 bits to the counter value and store updated value in memory.

7.6.1 Counter Manager Usage

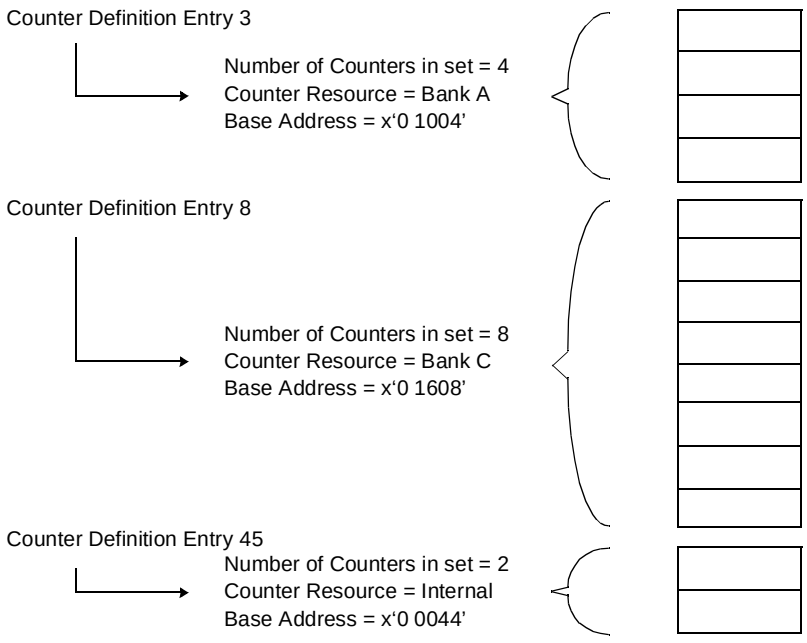
The Counter Manager manages various counters for the EPC. The EPC, Policy Manager, and the Ingress and Egress Flow Control have the ability to update these counters. The picocode accesses these counters for statistics, flow control actions, and policy decisions. Before a counter can be used, its counter definition entry must be configured. This entry defines where the counter is stored, how many counters are associated with this set, and the counter's size. The Counter Manager supports 256 different counter definition entries. The counter definition entry is written to the Counter Definition Table using the CAB. The entry format in shown in *Table 139*.

Table 139: Counter Definition Entry Format

Field	Bits	Definition
Reserved	31:30	Not used.
24/40	29	Flag to indicate 24/40 counter (1 = 24/40). If set, 64 / $\overline{32}$ must also be set to 1.
64 / $\overline{32}$	28	Flag to indicate a 64-bit counter (1 = 64) or a 32-bit counter (0 = 32).
Number of Counters	27:23	Number of counters within this counter set. Legal values: 1, 2, 4, 8, or 16
Counter Resource Location	22:20	Storage used for the counter block. 000 Bank A 001 Bank B 010 Bank C 011 Bank D 100 Internal Memory
Base Address	19:0	Base Address within the memory where the counter block starts.

The Counter Definition Entry describes a set of counters that have similar characteristics and are referenced from the picocode as a single group. The following figure shows several counter definition examples.

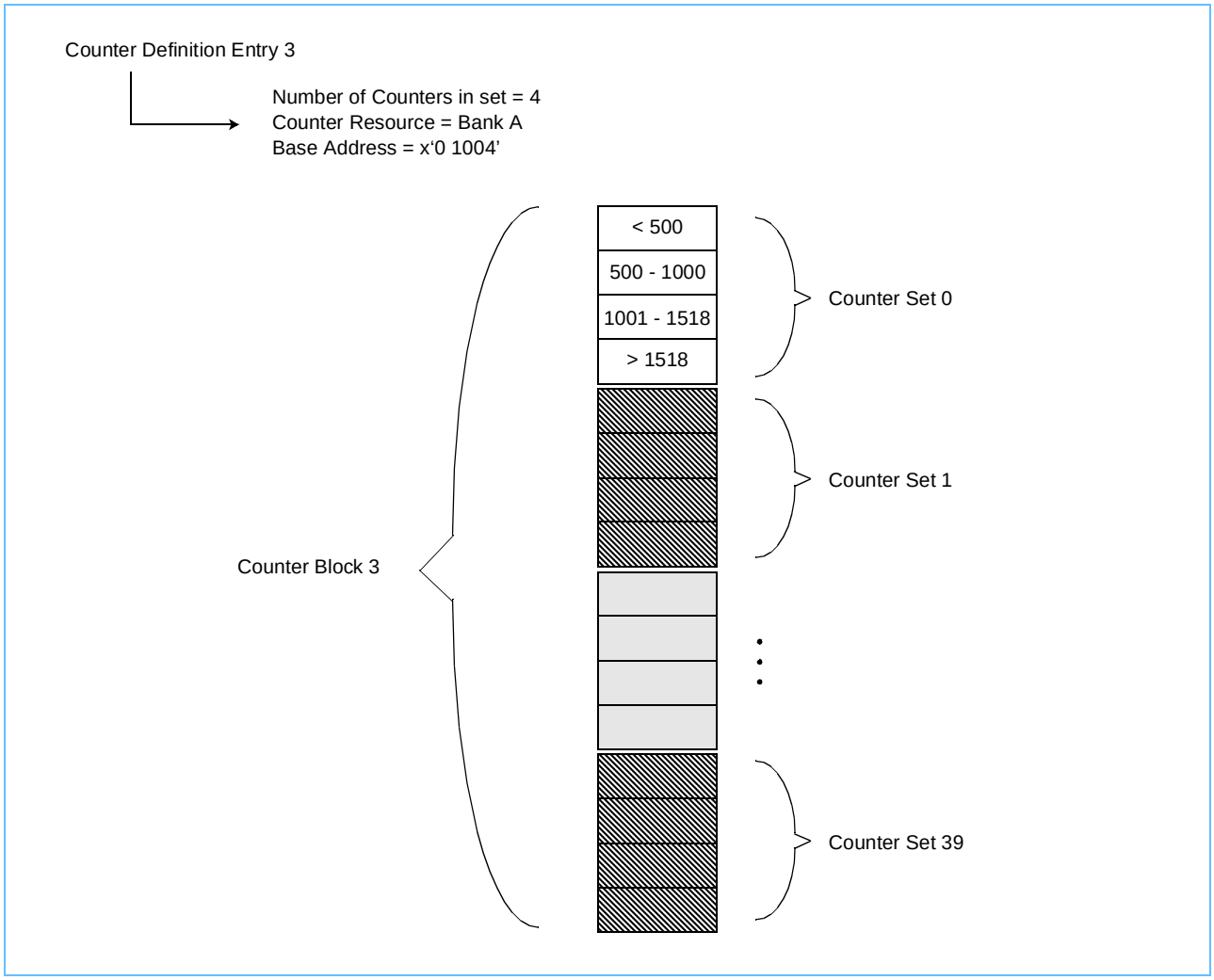
Figure 54: Counter Definition Entry



Each Counter Definition Entry can be used for a block of counter sets. The definition describes one counter set and the picocode can reference several consecutive counter sets using the same definition. For example, a counter set is defined as four counters: one for frames less than 500 bytes, one for frames between 500 and 1000 bytes, one for frames between 1001 and 1518 bytes, and the last one for frames greater than 1518. One Counter Definition Entry describes the counters and picocode can use the definition to reference 40 similar sets of these counters, e.g. one for each source port. Counter Set 0 is located at the Base Address

defined by the entry, Counter Set 1 is located at the next available address, and so on. *Figure 55* shows an example of counter blocks and sets.

Figure 55: Counter Blocks and Sets



In order to reference the correct counter, the requestor must pass the Counter Manager several parameters. These parameters are described in*Table 140*.

Table 140: Counter Manager Passed Parameters

Parameter	Bits	Definition
Counter Definition Table Index	8	Counter Definition Entry to use for this action
Counter Set Index	20	Set of counters to reference
Counter Number	4	Counter within the set to reference
Action	3	Action to perform on the counter Modify 000 Increment by 1 001 Add 16 bits to counter Read 100 Standard read 101 Read then Clear value Write 110 Write bits 15:0 of counter 111 Write bits 31:16 of counter All other code points are reserved.
Add/Write Value	16	Value to add to counter when Modify/Add selected Value to write to counter when Write selected
Flow Control Action (Counter Definition Table Index Offset)	2	Only used by Flow Control Interfaces 00 Standard Enqueue 01 Discard (resulting from the Transmit Probability table) 10 Tail Drop Discard 11 Reserved

The Counter Manager builds the actual counter address using the following algorithm:

Address = Base Address + (Counter Set Index * Number of Counters) + Counter Number

This address is used to access the counter within the appropriate memory (Bank A, Bank B, Bank C, Bank D, or Internal). When a 32-bit counter is accessed, the low order bit of the address selects which 32 bits of the 64-bit memory word are being used: 0 = bits 31:0, 1 = bits 63:32.

The Counter Manager supports a special mode of operation when used by the Ingress or Egress Flow Control. This mode allows “delayed increments” to occur based on flow control information. Both Flow Controls pass an additional parameter, Flow Control Action, that causes the Counter Manager to modify which Counter Definition Entry is used. A standard enqueue sent by either Flow Control logic causes the Counter Manager to use the Counter Definition Index that is passed with the request. A discard resulting from the Transmit Probability table (see the IBM PowerNP NP4GS3 Hardware Reference Manual, sections 3 and 5) causes the Counter Manager to access the Counter Definition Entry that is located at Index+1. A Tail Drop Discard causes the Counter Manager to access the Counter Definition Entry located at Index+2. Each type of flow control action uses a different counter definition.

When the Policy Manager requests use of the Counter Manager, it always uses a Counter Definition Table Index of 0. This location is reserved for use by the Policy Manager.

When accessing the internal memory, the Counter Manager can update these counters at a rate of one every 15 ns. The external DRAM rates are once each 150 ns or each 165 ns depending on DRAM cycle configuration (10 or 11 cycle windows).

The Counter Manager supports the following actions: read, read/clear, write/lower, write/upper, increment and add. The DPPUs can read any counter in the Counter Manager. The Flow Control and Policy Manager

logic do not perform counter reads. When a counter is read, the Counter Manager returns the read data (either 32 or 64 bits) with the acknowledge signal. The picocode can also clear the counter (set to zero) after the read is performed by passing the Read/Clear action. Counter writes of 16 bits are supported and either bits 15:0 or bits 31:16 of the selected counter are set to the write value with all other bits set to zero.

The Counter Manager also supports incrementing or adding to the selected counter. The add value can be up to 16 bits in size and will be added to the counter (either 32, 40, or 64 bits). When a 24/40 counter is incremented or added, the 24-bit portion of the counter is incremented and the 40-bit portion receives the increment/add action.

The Counter Manager supports the following maximum numbers of counters (actual number depends on size of DRAM used and the Counter Definition Table information):

- Up to 1 K 64-bit, or 2 K 32-bit Internal Counters, or some mix of 64 and 32-bit counters not to exceed 64 K bits total size.
- Up to 1 M 64-bit, or 1 M 32-bit External Counters per DRAM Bank, or some mix of 64 and 32-bit counters not to exceed 1 M total counters per bank.

8. Tree Search Engine

8.1 Overview

The Tree Search engine (TSE) is a hardware assist that performs table searches. Tables in the network processor are maintained as Patricia trees, with the termination of a search resulting in the address of a leaf page. The format of a leaf page or object is defined by the picocode; the object is placed into a control store, either internal (H0, H1) or external (Z0, D0-D3).

8.1.1 Addressing Control Store

References to the Control Store use a a 26-bit address as shown in *Table 141*. Each address contains a memory ID, a bank number, and an offset.

Table 141: Control Store Address Mapping for TSE References

26-bit Address Used to Reference the Control Store		
4 bits	2 bits	20 bits
Memory ID	Bank No.	Offset
Virtual Bank Address		

Table 142: CS Address Map and Use on page 220 provides addressing information and recommended uses for each memory that is supported by the CSA and that is accessible via the TSE. The memory Type provides access width and memory size information. Selection for D0 as either single or double wide is by configuration (*13.1.2 DRAM Parameter Register (DRAM_Parm)* on page 342).

Table 142: CS Address Map and Use

Memory Id	Bank No.	Memory Name	Type	Offset width (bits)	Recommended Use
0000	00	Null		20	
	01-11	Reserved			
0001	00	Z0	External ZBT SRAM	17 -18	Fast PSCB
	01-11	Reserved	512K x 36		
0010	00	H0	Internal SRAM 2K x 128	11	Fast Leaf pages
	01	H1	Internal SRAM 2K x 36	11	Fast internal SRAM
	10	Reserved			
	11	Reserved			
0011	00-11	Reserved			
0100 - 0111	00-11	Reserved			
1000	00-11	D0 banks A - D	DDR DRAM D0_Width=0 4 x 256K x 64 4 x 512K x 64 4 x 1M x 64 D0_Width=1 4 x 256K x 128 4 x 512K x 128 4 x 1M x 128	18 - 20	Single (D0_width=0) or double wide (D0_width=1) configuration. Leaf Pages
1001	00-11	D1 banks A - D	DDR DRAM	18 - 20	Leaf Pages
1010	00-11	D2 banks A - D	4 x 256K x 64		Leaf or Counter pages
1011	00-11	D3 banks A - D	4 x 512K x 64 4 x 1M x 64		DT entries and PSCB entries
1100	00-11	D6 banks A - D	DDR DRAM 4 x 1M x 64 4 x 2M x 64 4 x 4M x 64	20	PowerPC external memory TSE object access H=1, W=4, or 32 bytes Addresses x‘00 0000' through ‘0F FFFF'
1101					Addresses x‘10 0000' through ‘1F FFFF'
1110					x‘20 0000' through ‘2F FFFF'
1111					x‘30 0000' through ‘3F FFFF'

Note: DDR DRAM is specified as: Number of banks x number of entries x burst access width (in bits).

8.1.2 Control Store Use Restrictions

The following restrictions apply:

- When D2 is used by the Counter Manager, the last location (highest address) in each bank must not be assigned for any use.
- DT entries can be stored only in H1, Z0, or D3. Note that since LPM DT entries are 64 bits wide and DT entries must have a height=width =1, LPM DT entries are restricted to D3.
- PSCBs can be stored only in H1, Z0, or D3.
- Leaf pages can be stored only in H0, D0, D1, and D2.

8.1.3 Object Shapes

Object shapes specify how the Control Store stores an object such as a leaf or pattern search control block (PSCB). A leaf is a control block that contains the reference key as a reference pattern. The pattern uniquely identifies the leaf in a tree and contains the data needed by the application initiating a tree search. The data is application dependent and its size or memory requirements are defined by the LUDefTable entry for the tree. See *Table 144: Height, Width, and Offset Restrictions for TSE Objects on page 224* and *Table 156: LUDefTable Entry Definitions* on page 233 for details.

Shape is defined by two parameters, height and width. Objects small enough to fit within a single memory or bank location are defined as having a height and width of 1 (denoted by 1,1), and therefore do not require shaping. For example, both a 32-bit and a 48-bit object stored in a DDR SDRAM bank would have a shape of (1,1).

Table 143: DTEntry, PSCB, and Leaf Shaping

Object	Shape
DTEntry	Always has a shape of height = 1, width = 1
FM PSCB	Always has a shape of height = 1, width = 1
LPM PSCB	Has a shape of height = 1, width = 1 or height = 2, width = 1 depending on the memory in which the PSCB resides. A memory with a line width of at least 64 bits should be used with height = 1 and a memory of 36 bits should be used with height = 2.
Leaf	Can have any shape that is allowed by the memory in which the leaf is located -maximum of 512 bits.

When objects do not fit into a single memory or bank location, they have heights and/or widths > 1:

- Height denotes the number of consecutive address locations in which an object is stored. For example, if the height of an object = 4 and the Control Store address = A, the object is stored in locations A, A+1, A+2 and A+3.
- Width denotes the number of consecutive banks in which the object is stored. Width is always = 1 for objects stored in ZBT SRAM, and could be > 1 for objects in DDR SDRAM. For example, if the width of an object = 3, and its height = 1, the object is stored in three consecutive banks (the virtual bank address is incremented by 1).
- An offset increment can carry out into the bank address bits. This is not a supported use of the CS memory and table allocation algorithms must be defined to avoid this condition.

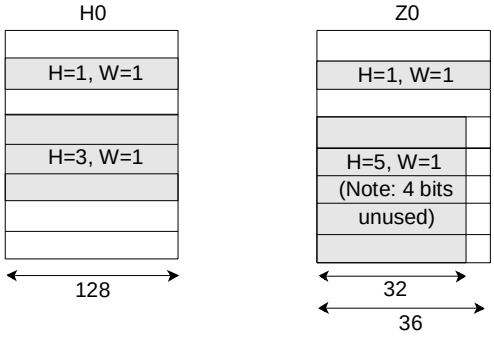
- For height and width, the hardware automatically reads the appropriate number of locations. From a picocode point of view, an object is an atomic unit of access. Restrictions to height, width, and offset are given in *Table 144: Height, Width, and Offset Restrictions for TSE Objects on page 224*. *Table 156: LUDefTable Entry Definitions* on page 233 specifies the leaf and PSCB shapes.

Figure 56: Example Shaping Dimensions on page 223 illustrates some example placement of objects with different shapes in control store. An object may span more than one DRAM as shown in examples (b) and (c) with the following restrictions:

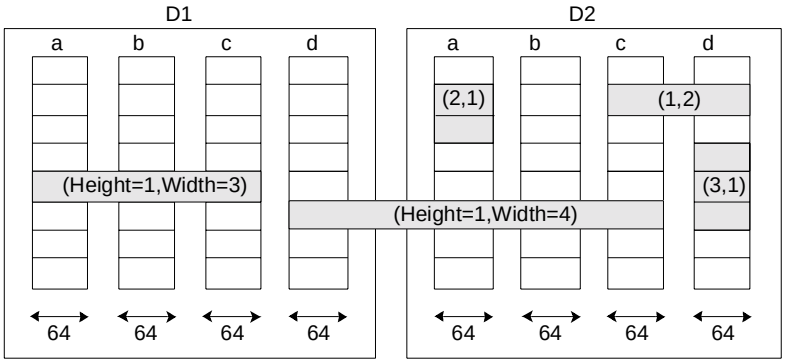
- An object may span D1 and D2
- When D0 is configured as a single wide (D0_Width=0), then an object may span D0 and D1.
- Objects may not span out of D0 when D0 is configured as a double wide (D0_Width=1).
- Objects may not span into or out of Z0, H0, H1, D3 or D6.

Figure 56: Example Shaping Dimensions

Example (a)



Example (b)



Example (c)

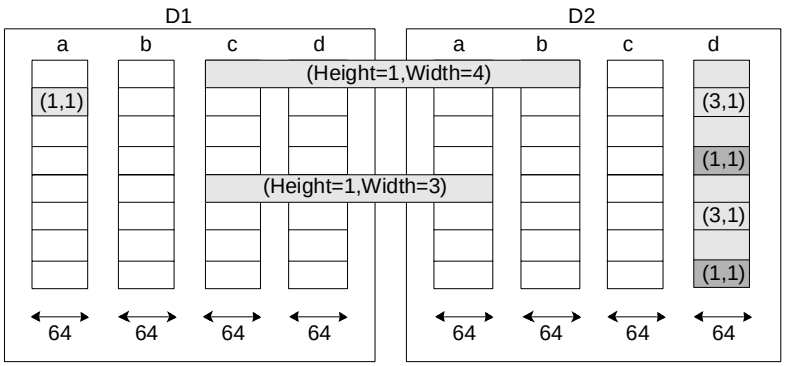


Table 144: Height, Width, and Offset Restrictions for TSE Objects

Memory	Height	Width	Total Object Size (bits)	Control Store Address Offset Must Be Divisible By
H0	1 2 3 4	1 1 1 1	128 256 384 512	1 2 4 4
H1	1 2 3 4 5 6 7 8	1 1 1 1 1 1 1 1	36 64 96 128 160 192 224 256	1 2 4 4 8 8 8 8
Z0	1 2 3 4 5 6 7 8	1 1 1 1 1 1 1 1	36 64 96 128 160 192 224 256	1 2 4 4 8 8 8 8
D0 (D0_Width=1; double wide configuration)	1 2 3 4 1 2 1 1	1 1 1 1 2 2 3 4	128 256 384 512 256 512 384 512	1 2 4 4 1 2 1 1
D0-D1-D2-D3-D6 (D0_Width=0; single wide configuration)	1 2 3 4 5 6 7 8 1 2 3 4 1 2 1 2	1 1 1 1 1 1 1 1 2 2 2 2 3 3 4 4	64 128 192 256 320 384 448 512 128 256 384 512 192 384 256 512	1 2 4 4 8 8 8 8 1 2 4 4 1 2 1 2

8.1.4 Illegal Memory Access

When the TSE uses an undefined MemoryID (that is., reserved) or an illegal memory shape, the TSE aborts the current command, returns a KO status, and sets an exception flag at bit 2 in interrupt class 3. The exception flag can be set to cause an interrupt by setting bit 2 of interrupt mask 3 to ‘1’. For debugging purposes, an exception can switch a programmable set of threads into single step mode.

8.1.5 Memory Range Checking

Memory range checking can flag access to a programmable range within MinBounds and MaxBounds. Memory range checking can be performed for any TSE Control Store accesses. When memory range checking is enabled, any Control Store address falling outside the range generates an exception (an exception does not stop TSE operation) and sets an exception flag at bit 1 in interrupt class 3 (TSM_Addr_Range_Violation). The exception flag can be set to cause an interrupt by setting bit 1 of interrupt mask 3 to '1'.

8.2 Trees and Tree Searches

The TSE uses trees to store and retrieve information. Tree searches, retrievals, inserts and deletes are performed according to a key that is similar to a MAC source address or a concatenation of an IP source and destination address. Information is stored in one or more leaves that contain the key as a reference pattern and, typically, contain aging and user information for forwarding purposes such as target blade and target port numbers.

To locate a leaf, a search algorithm processes input parameters that include the key and hashes the key. The algorithm then accesses a direct table (DT) and walks the tree through the PSCBs. There are three types of trees, each with its own search algorithm and tree-walk rules: full match (FM); longest prefix match (LPM); and software managed (SMT).

The data structure of FM and LPM trees is the Patricia tree. When a leaf is found, the leaf is the only candidate that can match the input key. A "compare-at-end" operation compares the input key with a reference pattern stored in the leaf to verify a match. Search results are "OK" when a match is found and "KO" in all other cases.

The data structure of SMT trees is similar to that of FM trees, but SMT trees can have multiple leaves that can be chained in a linked list. All leaves in the chain are checked against the input key until a match is found or the chain is exhausted. Search results are "OK" when a match is found and "KO" in all other cases.

Table 145: FM and LPM Tree Fixed Leaf Formats

Field Name	Byte Length	Description
NLARope	4	Leaf chaining pointer, aging and direct leaf information
Prefix_Len	1	Length of the pattern (in bits) for LPM only. Not used by TSE for FM trees and can be used by picocode
Pattern	2-18	Pattern to be compared with HashedKey. Always present. Length given by P1P2_max_size from *Table 156: LUDefTable Entry Definitions* on page 233
UserData	Variable	Under picocode control. For example, field can include one or more counters

Table 146: SMT Tree Fixed Leaf Formats

Field Name	Byte Length	Description
NLASMT	4	Leaf chaining pointer to chain leaves for SMT. Includes shape of chained leaf
Comp_Table_Index	1	Defines index in CompDefTable that defines compare between Pattern1, Pattern2, and HashedKey
Pattern1 and Pattern2	4-36	Contains Pattern1 and Pattern2 bitwise interleaved (even bits represent Pattern1 and odd bits represent Pattern2). That is, bit 0 of the field contains bit 0 of Pattern1, bit 1 contains bit 0 of pattern 2, etc. Length given by 2*P1P2_Max_Size from *Table 156: LUDefTable Entry Definitions* on page 233
UserData	Variable	Under picocode control. For example, field can include one or more counters

Table 147: Search Input Parameters

Parameter	Bit Length	Description
Key	192	Key must be stored in the Shared Memory Pool. All 192 bits must be set correctly (for Key Length shorter than 192, remaining bits in Shared Memory Pool must be set to 0).
Key Length	8	Contains Key Length minus 1 in bits.
LUDefIndex	8	Index into LUDefTable that points to an entry containing a full definition of the tree in which the search occurs. See section *8.2.5.1 The LUDefTable* on page 233.
LCBANr	1	Search results can be stored in either TSRx, as specified by TSEDPA. Leaf addresses are stored in LCBA0 or LCBA1 as specified by LCBANr. During a TSE search, picocode can access the other TSR to analyze the results of previous searches.
Color	16	For trees with color enabled, as specified in the LUDefTable, the contents of the color register are inserted into the key during hash operation. See section *8.2.1 Input Key and Color Register for FM and LPM Trees* on page 226 for an explanation of the process.

8.2.1 Input Key and Color Register for FM and LPM Trees

For FM and LPM trees, the input key is hashed into a HashedKey according to the algorithm specified in the LUDefTable. To minimize the depth of the tree that begins after the direct table, the hash function output is always a 192-bit number with a one-to-one correspondence to the original input key. Maximum output entropy is contained in the hash function's most significant bits. The N highest bits of the HashedKey register are used to calculate an index into the direct table, where N is determined by the definition of the DT (Direct Table) entry for the tree.

When colors are enabled for a tree, the 16-bit color register is appended as the LSB to the 176 MSBs from the 192-bit hashed output to produce the final 192-bit HashedKey. This occurs directly after the direct table. If the direct table contains 2^N entries, the 16-bit color value is inserted at bit position N. The hash function output and the inserted color value (when enabled) are stored in the HashedKey register. When colors are disabled, the 192-bit hash function is unmodified.

Colors can be used to share a single direct table among multiple independent trees. For example, color could indicate a VLANID in a MAC source address table. The input key would be the MAC SA and the color the VLANID (VLANID is 12 bits and 4 bits of the color would be unused, that is., set to 0). After the hash function, the pattern would be 48 + 16, or 64 bits. The color would be part of the pattern to distinguish MAC addresses of different VLANs.

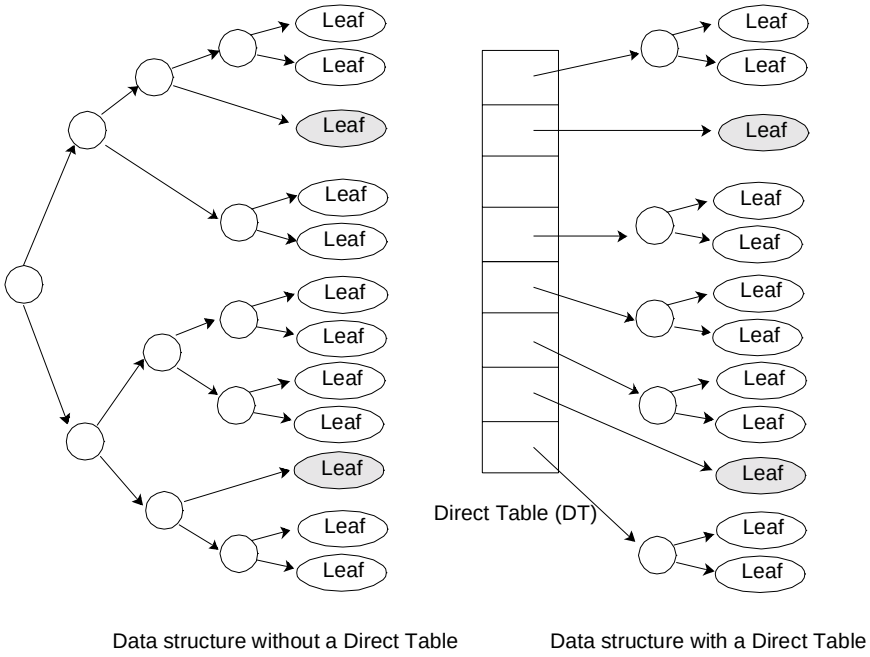
8.2.2 Input Key and Color Register for SMT Trees

For SMT trees, the input key is a 192-bit pattern and the color register is ignored. No hashing is performed.

8.2.3 Direct Table

A search starts when a DTEntry is read from the direct table. The read address is calculated from the N highest bits of the HashedKey and from the tree properties defined in the LUDefTable. The DTEntry can be represented as the root of a tree, with the actual tree data structure depending upon tree type. A Patricia tree data structure is used with FM trees. Extensions to the Patricia tree are used with LPMs and SMTs. Using a DT can reduce search time (PSCB access time). Increasing DT size is a trade-off between memory usage and search performance. When a single leaf is attached to a DTEntry, the read data includes a pointer to the leaf. When more than one leaf is attached, the read data defines the root of a tree. When the DTEntry is empty, no leaf information is attached.

Figure 57: Effects of Using a Direct Table



8.2.3.1 Pattern Search Control Blocks (PSCB)

A search begins after a DTEntry has been read if the DTEntry is neither empty nor contains a direct leaf. A tree walk search starts at the DTEntry and passes one or more PSCBs until a leaf is found. For an FM tree, the PSCB represents a node in the tree, or the starting point of two branches, “0” and “1”. Each PSCB is associated with a bit position “p” in the HashedKey. Bit p is the next bit to test (NBT) value stored in the previous PSCB or in the DTEntry. Leaves reachable from a PSCB through the 0 branch have a ‘0’ in bit p, and leaves reachable through the 1 branch have a ‘1’. Leaves reachable through either branch have patterns where bits 0..p-1 are identical, because pattern differences begin at bit p.

When an FM tree search encounters a PSCB, the TSE continues the tree walk on the 0 or 1 branch depending on the value of bit p. Thus, PSCBs are only inserted in the tree at positions where leaf patterns differ. This allows efficient search operations since the number of PSCBs, and thus the search performance, depends on the number of leaves in a tree, not on the length of the patterns.

8.2.3.2 Leaves and Compare-at-End Operation

The entire HashedKey is stored in the leaf as a reference pattern, not as the original input key. During a tree walk, only the HashedKey bits for which a PSCB exists are tested. When an FM leaf is found, its reference pattern must be compared to the full HashedKey to make sure all the bits match. If the leaf contains a chain pointer or NLA field to another leaf, the new leaf's reference pattern is compared to the HashedKey. Lacking a match or another NLA field, the search ends and the failure is indicated by a KO status. If the pattern matches, the original input key is checked. If that matches, the whole leaf page is returned to the network processor. If there is no match, the leaf page is returned with a no match message.

8.2.3.3 Cache

The direct table can be used as a cache to increase tree search performance since these trees are generally small and contain most likely entries. During a search, the TSE first determines whether the DT contains a leaf matching the HashedKey. If so, the leaf is returned, eliminating the need for a search. To the TSE, a cache lookup is identical to a normal search, that is., the input key is hashed into a HashedKey and the DT is accessed.

Caches are enabled in the LUDefTable on a per-tree basis. If a cache search uses LUDefTable entry I and the search ends with KO, another search using LUDefTable entry I+1 starts automatically. This allows multiple search chaining, although the full tree should be stored under LUDefTable entry I+1.

8.2.3.4 Cache Flag and NrPSCBs Registers

Picocode initiates insert and delete operations to and from the cache. Each search result stores information about the cache in the CacheFlag and NrPSCBs registers as shown in *Table 148*. Each register is divided into two sections, one for searches using TSR0 and the other for searches using TSR1.

Table 148: Cache Status Registers

Register	Bit Length	Description
CacheFlag(0)	1	CacheEmpty bit - set when cache search finds an empty DTEntry in cache DT
CacheFlag(1)	1	CacheLeafFound bit - set when cache search finds a leaf and cache search returns OK. When leaf found bit has been set, full search has not been performed.
CacheFlag(2)	1	CacheKO bit - set when cache search returns KO. When cache is empty, this bit is also set.
NrPSCBs	8	After any search, contains the number of PSCBs read during a tree walk. When a cache search finds no leaf and a full search starts, contains the number of PSCBs read during the search.

8.2.3.5 Cache Management

Cache management is performed using picocode. Cache inserts are controlled by inspecting the CacheFlag and NrPSCBs registers after each tree search. Inserts are treated like normal FM tree inserts, allowing the association of multiple leaves with a single DTEntry, because normal FM inserts create PSCBs to handle

multiple leaves. Inserts can also be done by writing directly to a DTEntry, although only using single leaves. Cache deletes use the tree aging mechanism whereby every N seconds all entries in the cache are deleted.

8.2.3.6 Search Output

The output of a search operation consists of the parameters listed in *Table 149*.

Table 149: Search Output Parameters

Parameter	Description
OK/KO flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation that is., leaf pattern matches HashedKey
TSRx	Leaf contents are stored in TSR0 or TSR1, depending on TSRNr
LCBA0 / 1	Leaf address is stored in LCBA0 / 1
CacheFlags(2)	CacheEmpty bit
CacheFlags(1)	CacheLeafFound bit
CacheFlags(0)	CacheKO bit
NrPSCBs	Number of PSCBs read during last search. Can be used as a criterion to insert cache entry

8.2.4 Tree Search Algorithms

The TSE provides hardware search operations for FM, LPM, and SMT trees. Software initializes and maintains trees. Leaves can be inserted into and removed from FM and LPM trees without control point function (CPF) intervention, permitting scalable configurations with CPF control when needed.

8.2.4.1 FM Trees

FM trees provide a mechanism to search tables with fixed size patterns, such as a Layer 2 Ethernet Unicast MAC tables which use fixed six-byte address patterns. Searches of FM trees are efficient because FM trees benefit from hash functions. The TSE offers multiple fixed hash functions that provide very low collision rates.

Each DTEntry is 36 bits wide and contains the formats listed in *Table 150*. PSCBs have the same structure as DTEntrys except they contain two PSCBLines, each of which can have one of the two pointer formats listed in this table*: Next Pointer Address (NPA) or Leaf Control Block Address (LCBA)*. The two PSCBLines are allocated consecutively in memory and are used as walking branches in the tree. The NBT value signifies which of the two PSCBLines is used.

Table 150: DTEntry and PSCBLine Formats

Format	Conditions	Valid in DTEntry?	Valid in PSCB?	Format (2 bits)	NPA/LCBA (26 bits)	NBT (8 bits)
Empty DTEntry	No leaves	Yes	No	00	0	0
Pointer to next PSCB	DTEntry contains pointer	Yes	Yes	00	NPA	NBT
Pointer to leaf	Single leaf associated with DTEntry; LCBA field contains pointer	Yes	Yes	01	LCBA	0

8.2.4.2 LPM Trees

LPM trees provide a mechanism to search tables with variable length patterns or prefixes such as a Layer 3

IP forwarding table where IP addresses can be full match host addresses or prefixes for network addresses. The CPF manages LPM trees with assistance from the GTH for inserting and removing leaf entries. LPM DTEntrys, each of which can contain a node address, an NPA, and an LCBA, differ from FM DTEntrys which cannot contain both a node and leaf address.

Each DTEntry is 64 bits wide and contains the formats listed in *Table 151: LPM DTEntry and PSCBLine Formats* on page 230. PSCBs have the same structure as DTEntrys except PSCBs contain two PSCBLines, each of which can have one of the three LCBA formats listed in the table. The two PSCBLines are allocated consecutively in memory and are used as walking branches in the tree. One of the PSCB lines may be empty, which is not allowed in FM PSCBs.

Table 151: LPM DTEntry and PSCBLine Formats

Format	Conditions	Valid in DTEntry?	Valid in PSCB?	Format (2 bits)	NPA (26 bits)	NBT (8 bits)	LCBA (26 bits)	Spare (2 bits)
Empty DTEntry	No leaves	Yes	Yes	00	0	0	0	0
LCBA not valid	DTEntry contains pointer to PSCB	Yes	Yes	00	NPA	NBT	0	0
LCBA valid; NPA/ NBT not valid	Single leaf associated with DTEntry; LCBA contains pointer to leaf; No pointer to next PSCB	Yes	Yes	01	0	0	LCBA	0
LCBA valid; NPA/ NBT valid	Single leaf associated with DTEntry; LCBA contains pointer to leaf; Pointer to next PSCB	Yes	Yes	01	NPA	NBT	LCBA	0

8.2.4.3 SMT Trees

SMT trees provide a mechanism to create trees that follow a CPF-defined search algorithm such as an IP quintuple filtering table containing IPSA, IPDA, source port, destination port, and protocol. SMT trees use the same PSCBs as FM trees, but only the first leaf following a PSCB is shaped by the *Table 156: LUDefTable Entry Definitions* on page 233. The following leaves in a leaf chain are shaped according to the 5 bits in the chaining pointer contained in the NLASMT leaf field (see *Table 152*). Unlike FM and LPM, SMT trees allow leaves to specify ranges, for instance, that a source port must be in the range of 100..110. SMT trees always contain two patterns of the same length in a leaf to define a comparison range. When the first leaf is found after a PSCB, a compare-at-end operation is performed. If OK, the search stops. If the comparison returns KO and the NLASMT field is non-zero, the next leaf is read and another compare-at-end operation is performed. This process continues until an “OK” is returned or until the NLASMT field is zero, which returns a KO.

Table 152: NLASMT Field Format

1 bit	2 bits	3 bits	26 bits
Reserved	Width of next leaf	Height of next leaf	NLA (Next Leaf Address)

8.2.4.4 Compare-at-End Operation

The input key and the two reference patterns stored in each leaf can be logically divided into multiple fields (see *Figure 58* on page 231). One of two comparisons can be performed on each field:

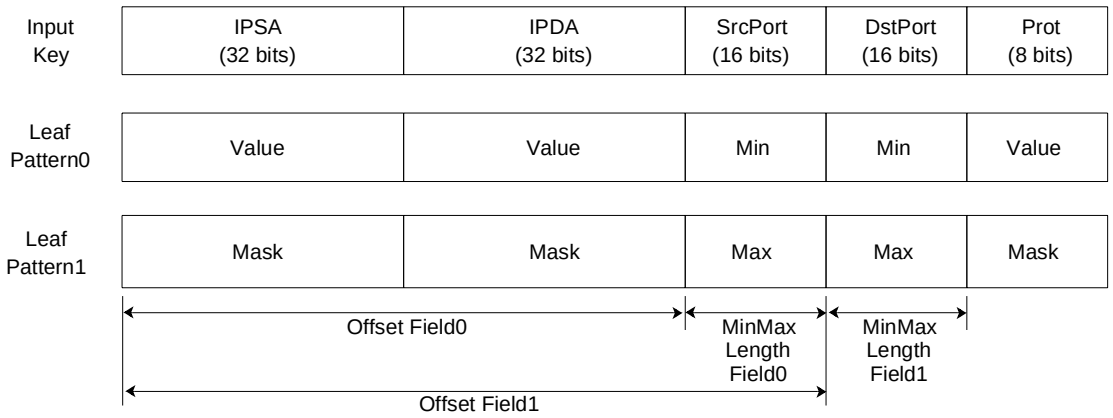
1. Compare under mask. The input key bits are compared to Pattern0 bits under a mask specified in Pattern1. A ‘1’ in the mask means the corresponding bit in the input key must equal the corresponding bit in Pattern0. A ‘0’ means the corresponding bit in the input key has no influence on the comparison. The

entire field matches only when all bits match, in which case the TSE returns OK.

2. Compare under range. The input key bits are treated as an integer and checked to determine whether the integer is within the range specified by Min and Max, inclusive. If so, the TSE returns OK, otherwise the TSE returns KO.

When all fields return OK, the entire compare-at-end returns OK. Otherwise, KO returns.

Figure 58: Example Input Key and Leaf Pattern Fields



Logical field definitions are specified in the compare definition table CompDefTable. *Table 153* shows the entry format for the CompDefTable. Fields are compare-under-mask unless specified by the CompDefTable. Each entry specifies one or two range comparisons, although multiple entries can specify more than two range comparisons. Each range comparison is defined by offset and length parameters. The offset, which is the position of the first bit in the field, must be at a 16-bit boundary and have a value of 0, 16, 32, 48, 64, 80, 96, 112, or 128. The length of the field must be 8, 16, 24, or 32 bits.

Table 153: CompDefTable Entry Format

Field	Range	Bit Length	Value
Range 1 Offset (R1_Offset)	1	8	Starting bit position for first range compare.
Range 1 Length (R1_Len)	1	8	Length and number of bits to be used as a part of range compare starting/ beginning from R1_Offset.
Range 2 Offset (R2_Offset)	2	8	Starting bit position for second range compare.
Range 2 Length (R2_Len)	2	8	Length and number of bits to be used as a part of range compare starting/ beginning from R2_Offset. This field is specified as the number of bits multiplied by 4.
Range 2 Valid (R2_Valid)	--	1	Range 2 Valid value. This field indicates whether or not the Range 2 Offset and Length fields are valid. 0 Range 1 Offset and Length valid, but Range 2 Offset and Length not valid. 1 Range 1 and Range 2 Offset and Length all valid.
Continue (Cont)	--	1	Continue indicator value. This field indicates whether the compare operation is continued in the next sequential entry of the table. 0 Comparison not continued 1 Comparison continued

In the input key shown in *Figure 58: Example Input Key and Leaf Pattern Fields* on page 231, the compare-

under-range for the Source Port (SrcPort) field would have Offset0 set to 64 and MinMaxLength0 set to 16. The compare-under-range for the Destination (DstPort) field would have Offset1 set to 80 and MinMaxLength1 set to 16. If more than two range comparisons are required, the Continue bit would be set to 1 so the next CompDefTable entry could be used for additional compare-under-range definitions.

The CompDefTable index used for tree comparisons is specified in the leaf Comp_Table_Index field (see *Table 146: SMT Tree Fixed Leaf Formats* on page 226). Each compare-under-range operation takes one clock cycle, so use as few compare-under-range operations as possible. If a range is a power of two, or 128-255, no compare-under-range is required since this range can be compared using compare-under-mask. When a compare-at-end fails and the SMTNLA leaf field is not 0, the TSE reads the next leaf and performs additional compare-at-end operations until the compare returns OK or until the SMTNLA is 0.

8.2.4.5 Ropes

As shown in the following figure, leaves in a tree can be linked together in a circular list called a rope. The first field in a leaf is the chaining pointer, or the NLARope. Picocode can “walk the rope,” or sequentially inspect all leaves in a rope. Ropes can be created by setting the NLARope_En bit to ‘1’ in the LUDefTable. See *Table 154: LUDefTable Rope Parameters* on page 232.

Figure 59: Rope Structure

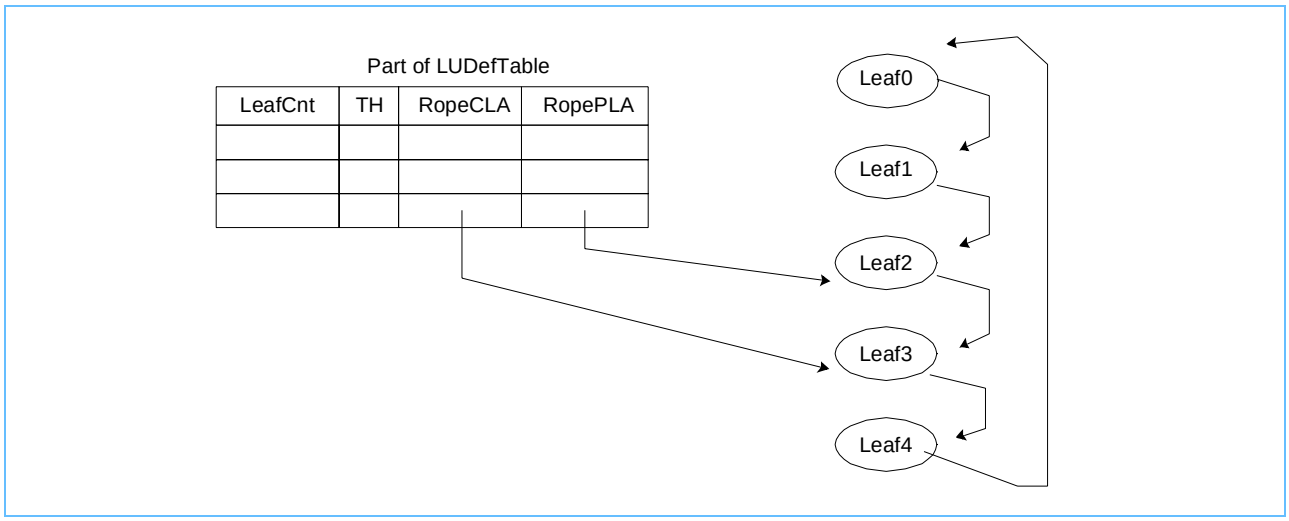


Table 154: LUDefTable Rope Parameters

Parameter	Description
RopeCLA	Current leaf address in the rope. All TSE instructions related to the rope such as RCLR, ARDL and TLIR (see section *8.2.8 GTH Hardware Assist Instructions* on page 247) relate to the leaf addressed by Rope-CLA.
RopePLA	Previous leaf address in the rope. Always the previous leaf if the rope is related to RopeCLA unless the rope contains no leaf or one leaf. The following condition is always true for trees with two or more leaves: RopePLA->NLARope == RopeCLA.
LeafCnt	Leaf count - number of leaves in the tree
LeafTh	Leaf threshold causes an exception to be generated when the LeafCnt exceeds the threshold

Leaf insertion is always done between a RopeCLA and RopePLA. After insertion, the RopePLA is unchanged and the RopeCLA points to the newly inserted leaf.

Leaf deletion is done two ways:

When the rope is not being walked, the rope is a single-chained linked list without a previous pointer. Leaves cannot be deleted without breaking the linked list. Setting the DeletePending bit postpones the deletion until the rope is walked again. A leaf is deleted by setting the DeletePending bit of the NLARope field. In this case, leaves are completely deleted and leaf pattern searches will return a KO.

When the rope is being walked and the DeletePending bit is set, the TSE deletes the leaf automatically.

8.2.4.6 Aging

Aging is enabled whenever a rope is created. The NLARope field contains one aging bit that is set to '1' when a leaf is created. When a leaf is found during a tree search and it matches the leaf pattern, the TSE sets the aging bit to '1' if it has not previously done so, and then writes this information to the Control Store. An aging function controlled by a timer or other device can walk the rope to delete leaves with aging bits set to '0' and then write the leaves to the Control Store with aging bits set to '0'. When no aging is desired, picocode should not alter the aging bit since bits set to '1' cannot be changed.

Table 155: NLARope Field Format

2 bits	1 bit	2 bits	1 bit	26 bits
Reserved	Aging Counter	Reserved	Delete Pending	NLA (Next Leaf Address)

8.2.5 Tree Configuration and Initialization

8.2.5.1 The LUDefTable

The lookup definition table (LUDefTable), an internal memory structure that contains 128 entries to define 128 trees, is the main structure that manages the Control Store. The table indicates in which memory (DDR-SDRAM, SRAM, or internal RAM) trees exist, whether caching is enabled, key and leaf sizes, and the search type to be performed. Each DTEntry contains two indexes to the Pattern Search and Leaf Free Queue (PSCB_Leaf_FQ). The first index defines which memory to use to create tree nodes, that is, where PSCBs are located. The second defines which memory to use to create leaves within a tree. When an entry is added to a tree, the memory required for the insert comes from the PSCB_Leaf_FQ and is returned when the entry is deleted. For SMT trees, an LU_Def_Tbl can be defined to match a value into a given range, but an index to an internal Compare Type Index must be given for the Compare Table (Cmp_Tbl).

Table 156: LUDefTable Entry Definitions

Field	Bit Length	Description
CacheEntry	1	0 Normal tree entry (not a cache entry) 1 A cache entry. When a search returns with KO, and CacheEntry = 1, TS instructions will restart using the next entry in the LUDefTable (that is., LUDefIndex + 1).
Tree_Type	2	Type of tree. 00 FM 01 LPM 10 SMT 11 Not used

Table 156: LUDefTable Entry Definitions (Continued)

Field	Bit Length	Description
Hash_Type	4	Hash type 0000 No Hash 0001 192-bit IP Hash 0010 192-bit MAC Hash 0011 192-bit Network Dispatch Hash 0100 No Hash 0101 48-bit MAC Swap 0110 60-bit MAC Swap 0111 192-bit SMT tree 1000 8-bit Hash 1001 12-bit Hash 1010 16-bit Hash
Color_En	1	Enable color register use by the hash function to construct HashedKey 0 Color register not used 1 Color register is used
P1P2_Max_Size	5	Maximum size of Pattern1 and Pattern2 in leaf - Indicates size in half words (that is., 16 bits) reserved for the “pattern” as a part of leaf in bytes. The maximum pattern size is 12, which represents 192 bits.
NLARope_En	1	Enable NLARope field use by leaf (enable aging) 0 Leaf does not use NLARope field 1 Leaf uses NLARope field
PSCB_FQ_Index	6	Defines the index of the PSCB free list
PSCB_Height	1	Height (part of the Shape) of a PSCBEntry. Width of a PSCB is always 1. 0 Height = 1 (FM ZBT SRAM, FM/LPM/SMT DDR SDRAM) 1 Height = 2 (LPM/SMT ZBT SRAM)
DT_Base_Addr	26	Provides DT base address.
DT_Size	4	Direct Table Size value - defines the number of DT entries within a memory bank. 0001 4 entries 0010 16 entries 0011 64 entries 0100 256 entries 0101 1 K entries 0110 4 K entries 0111 16 K entries 1000 64 K entries 1001 256 K entries 1010 1 M entries Others Reserved
Leaf_FQ_Index	6	Defines index of leaf free list
Leaf_Width	2	Leaf width
Leaf_Height	3	Leaf height
LUDef_State	2	LUDef state - used by Product Software/User Code to indicate current status of the entry. These bits do not affect Tree Search operation.

Table 156: LUDefTable Entry Definitions (Continued)

Field	Bit Length	Description
LUDef_Invalid	1	Indicates whether or not this entry is valid for normal Tree Search command or not. This bit blocks any tree search while the tree is being built or swapped. 0 Valid 1 Invalid When the LUDef Read request initiated by the Tree Search command finds this bit set to '1', it will re-read this LuDefEntry every 16 cycles until it is set to Valid. This halts the tree search for this particular thread.
Following fields are valid for GTH Only		
RopeCLA	26	Current leaf address for rope
RopePLA	26	Previous leaf address for rope
LeafCnt	26	Number of leaves in tree
LeafTh	10	Threshold for the number of leaves in a tree. If the LeafCnt is greater than this assigned threshold, then a class 0 interrupt is generated (bit 9 of the class 0 interrupt vector). The threshold is formed by the generation of two 26-bit numbers that are bit-wise ORed resulting in a threshold value that is compared to the LeafCnt: threshold(25:0) = TH9_5(25:0) OR TH4_0(25:0) TH9_5(25:0) = 0 when LeafTH(9:5)=0, otherwise TH9_5(25:0) = 2**(LeafTH(9:5)-1) TH4_0(25:0) = 0 when LeafTH(4:0)=0, otherwise TH4_0(25:0) = 2**(LeafTH(4:0)-1)

8.2.5.2 TSE Free Lists (TSE_FL)

The GTH has access to a set of 64 TSE free list control blocks, each defining a free list using the format shown in *Table 157*. Free lists typically chain unused PSCBs and leaves into a linked list, but can also be used by picocode to manage memory. The link pointer is stored in the Control Store at the address of the previous list object. Objects stored in different memories and with different shapes should be placed in different free lists. For example, a list of 64-bit PSCBs stored in both ZBT SRAM and internal RAM should have different entries. The GTH thread executes the TSENQFL and TSDQFL commands to enqueue and dequeue addresses on a free list. See section *8.2.8.3 Tree Search Enqueue Free List (TSENQFL)* on page 248 and section *8.2.8.4 Tree Search Dequeue Free List (TSDQFL)* on page 249.

A free list of N objects, each with shape of width = W, height = H and a start address of "A", is created by enqueueing address A, A+H, A+2H, A+3H, … A+(N-1)H in the free list. To prevent memory bottlenecks, free lists can also be created in a "sprayed" manner for objects contained in SDRAM. For example, when searching a large LPM tree with five PSCBs in a single bank, the bank must be accessed five times before reaching a leaf. When the PSCBs are sprayed among multiple banks, the number of accesses remains identical but accesses are to multiple banks, thus eliminating bottlenecks.

Table 157: Free List Entry Definition

Field	Bit Length	Description
Head	26	Free list head address in the control store.
Tail	26	Free list tail address in the control store.
QCnt	26	Number of entries in the free list.
Threshold	5	Threshold value for the free list control block entry.This field is iniitialized to 0. The threshold is determined as 2**Threshold. When the QCnt is less than the threshold, a Class 0 interrupt (bit 8) is generated.

8.2.6 TSE Registers and Register Map

Table 158: TSE Scalar Registers for GTH Only

Name	Read/ Write	Hex Address	Bit Length	Description
Color	R/W	00	16	Color - see section *8.2.1 Input Key and Color Register for FM and LPM Trees* on page 226 and section *8.2.2 Input Key and Color Register for SMT Trees* on page 227.
LCBA0	R/W	02	26	Leaf Control Block Address 0 - typically contains the Control Store address of the leaf in TSR0, but is also used as an address register for various TSE commands.
LCBA1	R/W	03	26	Leaf Control Block Address 1 - typically contains the Control Store address of the leaf in TSR1, but is also used as an address register for various TSE commands.
DTA_Addr	R/W	04	26	DTEntry Address - valid after a hash has been performed.
DTA_Shape	R/W	05	5	Shape of a DTEntry - always (1,1) when direct leaves are disabled. Equals the leaf shape as defined in LUDefTable when direct leaves are enabled. Always set to '00000'
HashedKeyLen	R/W	06	8	Pattern length minus 1 in HashedKey
CacheFlags	R	07	3	See section *8.2.3.3 Cache* on page 228
NrPSCBs	R	08	8	See section *8.2.3.3 Cache* on page 228
HashedKey	R/W	0A-0F	192	Contains HashedKey
HashedKey 191_160	R/W	0A	32	Bits 191..160 of HashedKey
HashedKey 159_128	R/W	0B	32	Bits 159..128 of HashedKey
HashedKey 127_96	R/W	0C	32	Bits 127..96 of HashedKey
HashedKey 95_64	R/W	0D	32	Bits 95..64 of HashedKey
HashedKey 63_32	R/W	0E	32	Bits 63..32 of HashedKey
HashedKey 31_0	R/W	0F	32	Bits 31..0 of HashedKey
LUDefCopy	R	10-12	96	Contains LUDefTable index in use and a copy of the following LUDefTablefields: CacheEntry, Tree_Type, hash_type, color_en, DT_size, DT_base_addr, DT_interleaf, DirectLeafEn, PSCB_Height, Leaf_Width, Leaf_Height, CompIndex, P1P2_max_size, NPARope_en, NPASMT_en, CompIndex_en
LUDefCopy 95_64	R	10	32	Bits 95..88 = x'0000 0000' Bits 87..80 = LuDef Index = From which location LuDefCopy content was read Bits 79..64 of LUDefCopy
LUDefCopy 63_32	R	11	32	Bits 63..32 of LUDefCopy
LUDefCopy 31_0	R	12	32	Bits 31..0 of LUDefCopy

Table 159: TSE Array Registers for All GxH

Name	Read/ Write	Starting Hex Address	Ending Hex Address	Bit Length	Description
TSR0	R/W	32	47	2048	Tree Search Result Area 0 Note: Portion of the data overlaps with TSR1
TSR1	R/W	40	47	1024	Tree Search Result Area 1 Note: Portion of the data overlaps with TSR0
TSR2	R/W	48	63	2048	Tree Search Result Area 2 Note: Portion of the data overlaps with TSR3
TSR3	R/W	56	63	1024	Tree Search Result Area 3 Note: Portion of the data overlaps with TSR2

Note: In this table, starting and ending Address represents the offset for the given thread's starting address in the Shared Memory Pool.

Table 160: TSE Registers for GTH (Tree Management)

Name	Read/ Write	Hex Address	Bit Length	Description
PatBit_TSR0	R/W	1E	1	Contains one bit from the pattern stored in TSR0. Set by TRS0PAT command
DistPosReg	R	1F	8	Contains result of DISTPOS command
LURopeCopyTH	R	13	10	Contains copy of LeafTH field of LUDefTable
LURopeCopyQCnt	R	14	26	Contains copy of LeafCnt field of LUDefTable
LURopeCopyPrev	R	15	26	Contains copy of RopePLA field of LUDefTable
LURopeCopyCurr	R	16	26	Contains copy of RopeCLA field of LUDefTable

Table 161: TSE Scalar Registers for GDH and GTH

Name	Read/ Write	Hex Address	Bit Length	Description
LCBA0	R/W	02	31	Leaf Control Block Address 0 - typically contains the Control Store address of the leaf in TSRx, but is also used as an address register for various TSE commands. Bits 30:26 The Leaf Control Block Address Shape which is used by CMPEND instruction only. Bits 25:0 Leaf Control Block Address
LCBA1	R/W	03	31	Leaf Control Block Address 1 - typically contains the Control Store address of the leaf in TSRx, but is also used as an address register for various TSE commands. Bits 30:26 The Leaf Control Block Address Shape which is used by CMPEND instruction only. Bits 25:0 Leaf Control Block Address
CacheFlags	R	07	3	See section *8.2.3.3 Cache* on page 228
NrPSCBs	R	08	8	See section *8.2.3.3 Cache* on page 228

Table 162: PSCB Register Format

Field	Bit Length	Control Store Address for PSCB
NPA0	26	Next PSCB address - pointer to next PSCB in tree for PSCB part 0
NBT0	8	Next bit to test for PSCB part 0
LCBA0	26	Leaf control block address: Pointer to leaf for PSCB part 0
NPA1	26	Next PSCB address - Pointer to next PSCB in tree for PSCB part 1
NBT1	8	Next bit to test for PSCB part 1
LCBA1	26	Leaf control block address - Pointer to leaf for PSCB part 1
Index	8	Index of current PSCB physically stored in previous PSCB
PatBit	1	Value of HashedKey[Index] based on value of index field in PSCB register

Table 163: TSE GTH Indirect Registers

Indirect Register	Bit Length	Description	Notes
PSCBx.NPA_HK	26	Selects either the NPA0 field from PSCBx or the NPA1 field depending on value of register PSCBx.Index	1, 2, 3
PSCBx.NPA_TSR0	26	Selects either the NPA0 field from PSCBx or the NPA1 field depending on value of register PatBit_TSR0. (Register PatBit_TSR0 must have been initialized previously using TSR0PAT command)	1, 2, 4
PSCBx.NBT_HK	8	Selects either the NBT0 field from PSCBx or the NBT1 field depending on value of register PSCBx.Index	1, 2, 3
PSCBx.NBT_TSR0	8	Selects either the NBT0 field from PSCBx or the NBT1 field depending on value of register PatBit_TSR0	1, 2, 4
PSCBx.LCBA_HK	26	Selects either the LCBA0 field from PSCBx or the LCBA1 field depending on value of register PSCBx.Index	1, 2, 3
PSCBx.LCBA_TSR0	26	Selects either the LCBA0 field from PSCBx or the LCBA1 field, depending on value of register PatBit_TSR0	1, 2, 4
PSCBx.NotNPA_HK	26	Selects either the NPA0 field from PSCBx or the NPA1 field depending on inverse value of register PSCBx.Index	1, 2, 3
PSCBx.NotNPA_TSR0	26	Selects either the NPA0 field from PSCBx or the NPA1 field depending on inverse value of register PatBit_TSR0	1, 2, 4
PSCBx.NotNBT_HK	8	Selects either the NBT0 field from PSCBx or the NBT1 field depending on inverse value of register PSCBx.Index	1, 2, 3
PSCBx.NotNBT_TSR0	8	Selects either the NBT0 field from PSCBx or the NBT1 field depending on inverse value of register PatBit_TSR0	1, 2, 4
PSCBx.NotLCBA_HK	26	Selects either the LCBA0 field from PSCBx or the LCBA1 field depending on inverse value of register PSCBx.Index	1, 2, 3
PSCBx.NotLCBA_TSR0	26	Selects either the LCBA0 field from PSCBx or the LCBA1 field depending on inverse value of register PatBit_TSR0	1, 2, 4

1. x must equal 0, 1, or 2.
2. The Indirect Registers of the TSE select, via dedicated hardware assist, one of the TSE registers listed in section *8.2.6 TSE Registers and Register Map* on page 236. The Indirect Registers appear in the TSE Register Map with a unique register number.
3. PSCBx.Index points to a specific bit in the HashedKey. The bit's value determines whether the 0 or 1 part of PSCBx will be read or written.
4. Value of PatBit_TSR0 determines whether the 0 or 1 part of PSCBx will be read or written.

Table 164: Address Map for PSCB0-2 Registers in GTH

PSCBx	Read/Write	PSCB0	PSCB1	PSCB2	Size
NPA0	R/W	80	A0	C0	26
NBT0	R/W	81	A1	C1	8
LCBA0	R/W	82	A2	C2	26
NPA1	R/W	84	A4	C4	26
NBT1	R/W	85	A5	C5	8
LCBA1	R/W	86	A6	C6	26
Addr	R/W	88	A8	C8	26
Index	R/W	89	A9	C9	8
PatBit	R	8B	AB	CB	1
NPA_HK	R/W	90	B0	D0	26
NBT_HK	R/W	91	B1	D1	8
LCBA_HK	R/W	92	B2	D2	26
NotNPA_HK	R/W	94	B4	D4	26
NotNBT_HK	R/W	95	B5	D5	8
NotLCBA_HK	R/W	96	B6	D6	26
NPA_TSR0	R/W	98	B8	D8	26
NBT_TSR0	R/W	99	B9	D9	8
LCBA_TSR0	R/W	9A	BA	DA	26
NotNPA_TSR0	R/W	9C	BC	DC	26
NotNBT_TSR0	R/W	9D	BD	DD	8
NotLCBA_TSR0	R/W	9E	BE	DE	26

8.2.7 TSE Instructions

Table 165: General TSE Instructions

Opcode	Command	Detail Section
0	Null	
1	TS_FM	*8.2.7.1 FM Tree Search (TS_FM)* on page 240
2	TS_LPM	*8.2.7.2 LPM Tree Search (TS_LPM)* on page 241
3	TS_SMT	*8.2.7.3 SMT Tree Search (TS_SMT)* on page 242
4	MRD	*8.2.7.4 Memory Read (MRD)* on page 243
5	MWR	*8.2.7.5 Memory Write (MWR)* on page 244
6	HK	*8.2.7.6 Hash Key (HK)* on page 244
7	RDLUDEF	*8.2.7.7 Read LUDefTable (RDLUDEF)* on page 245
8	COMPEND	*8.2.7.8 Compare-at-End (COMPEND)* on page 245
9 to 15	Reserved	
Note: Commands can be executed by all GxHs with threads.		

8.2.7.1 FM Tree Search (TS_FM)

Table 166: FM Tree Search Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable that controls the search.
LCBANr	1	Imm16(0)	Imm12(0)	0 search results are stored in TSRx/LCBA0. 1 search results are stored in TSRx/LCBA1.
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	TSE Thread Shared Memory Pool Address - stores location of Key, KeyLength, and Color and determines Leaf destination. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
Key	192	Shared Memory Pool		Key - pattern to be searched. Must be initialized before search. Located in Shared Memory Pool.
KeyLength	8	Shared Memory Pool		KeyLength - length of pattern minus 1 in key. Must be initialized before search (done automatically using key instructions) Located in Shared Memory Pool.
Color	16	Shared Memory Pool		Color - used only when enabled in LUDefTable. Must be initialized before search. Located in Shared Memory Poo.l
Following is available for GTH only.				
UseLUDefCopyReg	1	Imm16(13)	Imm12(5)	Enables TSE read of LUDefCopy register. Can save clock cycles, especially when RDLUDEF is executed asynchronously with the picocode that sets the key. 0 TSE reads LUDefTable. 1 TSE does not read the LUDefTable and uses information contained in LUDefCopy register. Assumes LUDefTable was read previously using RDLUDEF.
LUDefCopy	96	Register		Input only when UseLUDefCopyReg is '1'.

Table 167: FM Tree Search Results (TSR) Output

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation. 1 OK: Successful Operation.
TSRx	512	Shared Memory Pool	When OK/KO is '1', leaf is read and stored in TSRx. TSRx is mapped into the Shared Memory pool, at an offset of 1 QW past the starting QW indicated by the input TSEDPA parameter. That is, the Shared Memory Pool QW location = TSEDPA*4 + 1
LCBA0 / 1	26	Register	When OK/KO is '1', leaf address is stored in LCBA0 / 1.
CacheFlags	3	Register	See section *8.2.3.3 Cache* on page 228.
NrPSCBs	8	Register	See section *8.2.3.3 Cache* on page 228.
Following is available for GTH only.			
LUDefCopy	96	Register	Output only when UseLUDefCopyReg is '0'. Set to contents of LUDefTable at entry pointed to by LUDefIndex.

8.2.7.2 LPM Tree Search (TS_LPM)

Table 168: LPM Tree Search Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable used to control the search
LCBANr	1	Imm16(0)	Imm12(0)	0 search results are stored in TSRx/LCBA0 1 search results are stored in TSRx/LCBA1
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
Key	192	Shared Memory Pool		Key - pattern to be searched. Must be initialized before search. Located in Shared Memory Pool.
KeyLength	8	Shared Memory Pool		KeyLength - length of pattern minus 1 in key. Must be initialized before search (done automatically using key instructions). Located in Shared Memory Pool.
Color	16	Shared Memory Pool		Color - used only when enabled in LUDefTable. Must be initialized before search. Located in Shared Memory Pool.
Following is available for GTH only.				
UseLUDefCopyReg	1	Imm16(13)	Imm12(5)	Enables TSE read of LUDefCopy register Can save clock cycles, especially when RDLUDEF is executed asynchronously with the picocode that sets the key. 0 TSE reads LUDefTable 1 TSE does not read the LUDefTable and uses information contained in LUDefCopy register. Assumes LUDefTable was read previously using RDLUDEF.
LUDefCopy	96	Register		Input only when UseLUDefCopyReg is '1'. Set to contents of LUDefTable at entry given by LUDefIndex

Table 169: LPM Tree Search Results (TSR) Output

Result	Bits	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
TSRx	512	Shared Memory Pool	When OK/KO is '1', leaf is read and stored in TSRx TSRx is mapped into the Shared Memory pool, at an offset of 1 QW past the starting QW indicated by the input TSEDPA parameter. That is, Shared Memory Pool QW location = TSEDPA*4 + 1
LCBA0 / 1	26	Register	When OK/KO is '1', leaf address is stored in LCBA0 / 1
CacheFlags	3	Register	See section *8.2.3.3 Cache* on page 228
NrPSCBs	8	Register	See section *8.2.3.3 Cache* on page 228
The following is available for GTH only.			
LUDefCopy	96	Register	Output only when UseLUDefCopyReg is '0'. Set to contents of LUDefTable at entry given by LUDefIndex

8.2.7.3 SMT Tree Search (TS_SMT)

Table 170: SMT Tree Search Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable used to control the search
LCBANr	1	Imm16(0)	Imm12(0)	0 search results are stored in TSRx/LCBA0 1 search results are stored in TSRx/LCBA1
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
Key	192	Shared Memory Pool		Key - pattern to be searched. Must be initialized before search. Located in Shared Memory Pool.
KeyLength	8	Shared Memory Pool		KeyLength - length of pattern minus 1 in key. Must be initialized before search (done automatically using key instructions). Located in Shared Memory Pool.
Color	16	Shared Memory Pool		Color - used only when enabled in LUDefTable. Must be initialized before search. Located in Shared Memory Pool.
Following is available for GTH only.				
UseLUDefCopyReg	1	Imm16(13)	Imm12(5)	Enables TSE read of LUDefCopy register Can save clock cycles, especially when RDLUDEF is executed asynchronously with the picocode that sets the key. 0 TSE reads LUDefTable 1 TSE does not read the LUDefTable and uses information contained in LUDefCopy register. Assumes LUDefTable was read previously using RDLUDEF.
LUDefCopy	96	Register		Input only when UseLUDefCopyReg is '1'. Set to contents of LUDefTable at entry given by LUDefIndex.

Table 171: SMT Tree Search Results (TSR) Output

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
TSRx	512	Shared Memory Pool	When OK is '1', leaf is read and stored in TSRx TSRx is mapped into the Shared Memory pool, at an offset of 1 QW past the starting QW indicated by the input TSEDPA parameter. That is, Shared Memory Pool QW location = TSEDPA*4 + 1
LCBA0 / 1	26	Register	When OK is '1', leaf address is stored in LCBA0 / 1
CacheFlags	3	Register	See section *8.2.3.3 Cache* on page 228
NrPSCBs	8	Register	See section *8.2.3.3 Cache* on page 228
Following are available for GTH only.			
LUDefCopy	96	Register	An output only when UseLUDefCopyReg is '0'. Set to contents of LUDefTable at entry given by LUDefIndex

8.2.7.4 Memory Read (MRD)

The memory read command provides direct read capability from any location in the Control Store. LCBA0 / 1 provide the full read address. Shape is provided by the LUDefTable (for Leaf or PSCB) or directly as part of the command field for the object. The content to be read is stored in TSRx.

Table 172: Memory Read Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
ShapeCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct shape 10 PSCB shape from LUDefTable 11 Leaf shape from LUDefTable
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	LUDefTable entry used to read shape information. Valid only when ShapeCtrl is '10' or '11'.
Width	2	Imm16(9..8)	GPR(4..3)	Width of object to be read. Valid only when ShapeCtrl is '00'.
Height	3	Imm16(7..5)	GPR(2..0)	Height of object to be read. Valid only when ShapeCtrl is '00'.
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.
LCBANr	1	Imm16(0)	Imm12(0)	0 Address to be read is LCBA0 1 Address to be read is LCBA1
LCBA0 / 1	26	Register		Address to be read

Table 173: Memory Read Output Results

Result	Bit Length	Source	Description
TSRx	512	Shared Memory Pool	TSRx is mapped into the Shared Memory pool, starting at the QW indicated by the input TSEDPA parameter for a length of 4 QW.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.7.5 Memory Write (MWR)

The memory write command provides direct write capability to any location in the Control Store. LCBA0 / 1 provide the full write address. Shape is provided by the LUDefTable (for Leaf or PSCB) or directly as part of the command field for the object. The content to be written is stored in TSRx.

Table 174: Memory Write Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
ShapeCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct Shape 10 PSCB shape from LUDefTable 11 Leaf shape from LUDefTable
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	LUDefTable entry used to read shape information. Valid only when ShapeCtrl is '10' or '11'.
Width	2	Imm16(9..8)	GPR(4..3)	Width of object to be read. Valid only when ShapeCtrl is '00'.
Height	3	Imm16(7..5)	GPR(2..0)	Height of object to be read. Valid only when ShapeCtrl is '00'.
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.
LCBANr	1	Imm16(0)	Imm12(0)	0 Address to be written is LCBA0 1 Address to be written is LCBA1
LCBA0 / 1	26	Register		Address to be written
TSRx	512	Shared Memory Pool		Data to be written. TSRx is mapped into the Shared Memory pool, starting at the QW indicated by the input TSEDPA parameter for a length of 4 QW

8.2.7.6 Hash Key (HK)

Table 175: Hash Key Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable containing hash type
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
Direct_ HashType_En	1	Imm16(0)	Imm12(0)	Enable Direct HashType definition 0 HashType defined by LUDefEntry 1 HashType defined via command
Direct_ HashType	4	Imm16(8..5)	GPR(3..0)	Use defined Hash Type for Hashing. Valid when Direct_HashType_En = 1
Key	192	Shared Memory Pool		Provides pattern to be searched. Must be initialized before search. Located in the Shared Memory Pool.
KeyLen	8	Shared Memory Pool		Defines length of pattern minus 1 in key. Must be initialized before search. Located in the Shared Memory Pool.
Color	16	Shared Memory Pool		Must be initialized before search - used only when enabled in LUDefTable. Located in the Shared Memory Pool. Invalid when Direct_HashType_En is set to value '1'.

Table 176: Hash Key Output Results

Result	Bit Length	Source	Description
HashedKeyReg	192	Shared Memory Pool	Hashed key register - contains the HashedKey (including color when enabled in LUDefTable) according to section *8.2.1 Input Key and Color Register for FM and LPM Trees* on page 226 and section *8.2.2 Input Key and Color Register for SMT Trees* on page 227. Hash function is defined in the LUDefTable. Stored in the Shared Memory Pool.
HashedKeyLen	8	Shared Memory Pool	Hashed key length - contains the length of pattern minus 1 in HashedKeyReg. Stored in the Shared Memory Pool.
DTA	26	Shared Memory Pool	DTEntry Address. Stored in Shared Memory Pool. Note: This is not valid when Direct_HashType_En is set to value '1'.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
Following is available for GTH only.			
LUDefCopy	96	Register	Set to contents of LUDefTable at entry given by LUDefIndex.

8.2.7.7 Read LUDefTable (RDLUDEF)

RDLUDEF reads LUDefTable at a specified entry and stores the result in the LUDefCopy register. The TSE can read LUDefTable while picocode builds a key because *RDLUDEF* is executed asynchronously. Once the key is ready, the tree search execution can be executed with the UseLUDefCopyReg flag set to '1'.

Table 177: RDLUDEF Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.

Table 178: RDLUDEF Output Results

Result	Bit Length	Source	Description
LUDefCopy	96	Shared Memory Pool	Set to contents of LUDefTable at entry given by LUDefIndex and stored in Shared Memory Pool. The entry is placed into two QW starting at the QW indicated by the TSEDPA. The entry is right-justified with the most significant bits padded with 0. For GTH, the scalar register will also have content of LUDefTable at the entry.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.7.8 Compare-at-End (COMPEND)

COMPEND performs a compare-at-end operation for SMT trees. After a tree search command has performed a full search including a COMPEND, picocode can obtain a pointer to another leaf chain from the leaf and start another COMPEND on the leaf chain. The first leaf chain could contain leaves with filtering informa-

tion, and the second could contain leaves with quality of service information. COMPEND should be used only after a tree search command.

Table 179: COMPEND Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12:5)	GPR(7:0)	Defines the entry in the LuDefTable.
LCBNANr	1	Imm16(0)	Imm12(0)	0 Search results are stored in TSRx/LCBA0 1 Search results are stored in TSRx/LCBA1
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
LCBA0 / 1	31	Register		Start address of leaf chain and its shape. Bits: 30:29 Leaf Width 28:26 Leaf Height 25:0 Leaf Address **Note:** A Valid Leaf Width and Leaf Height must be provided with the Leaf Address.
HashedKey	192	Shared Memory Pool		Output of previous tree search command located in the Shared Memory Pool at an offset of 2 and 3 QW from the QW indicated by the TSEDPA.
HashedKeyLen	8	Shared Memory Pool		Output of previous tree search command located in the Shared Memory Pool at the QW indicated by the TSEDPA

Table 180: COMPEND Output Results

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
TSRx	512	Shared Memory Pool	When OK is '1', leaf is read and stored in TSRx When OK is '0', last leaf of chain is stored in TSRx TSRx is mapped into the Shared Memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA parameter for a length of 4 QW.
LCBA0 / 1	26	Register	When OK is '1', leaf address is stored in LCBA0 / 1 When OK is '0', leaf address of last leaf in chain is stored in LCBA0 / 1 **Note:** Shape for the corresponding leaf will not be valid when LCABA0/1 is read.

8.2.8 GTH Hardware Assist Instructions

Table 181: General GTH Instructions

Opcode	Command	Detail Section
16	HK_GTH	*8.2.8.1 Hash Key GTH (HK_GTH)* on page 247
17	RDLUDEF_GTH	*8.2.8.2 Read LUDefTable GTH (RDLUDEF GTH)* on page 248
18	TSENQFL	*8.2.8.3 Tree Search Enqueue Free List (TSENQFL)* on page 248
19	TSDQFL	*8.2.8.4 Tree Search Dequeue Free List (TSDQFL)* on page 249
20	RCLR	*8.2.8.5 Read Current Leaf from Rope (RCLR)* on page 250
21	ARDL	*8.2.8.6 Advance Rope with Optional Delete Leaf (ARDL)* on page 251
22	TLIR	*8.2.8.7 Tree Leaf Insert Rope (TLIR)* on page 251
23	Reserved	
24	CLRPSCB	*8.2.8.8 Clear PSCB (CLRPSCB)* on page 252
25	RDPSCB	*8.2.8.9 Read PSCB (RDPSCB)* on page 252
26	WRPSCB	*8.2.8.10 Write PSCB (WRPSCB)* on page 253
27	PUSHPSCB	*8.2.8.11 Push PSCB (PUSHPSCB)* on page 254
28	DISTPOS	*8.2.8.12 Distinguish (DISTPOS)* on page 254
29	TSR0PAT	*8.2.8.13 TSR0 Pattern (TSR0PAT)* on page 255
30	PAT2DTA	*8.2.8.14 Pattern 2DTA (PAT2DTA)* on page 255
31	Reserved	

Note: The instructions listed in *Table 165: General TSE Instructions* on page 240 can only be executed by the GTH

8.2.8.1 Hash Key GTH (HK_GTH)

Table 182: Hash Key GTH Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable containing hash type
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.
Direct_ HashType_En	1	Imm16(0)	Imm12(0)	Enable Direct HashType definition 0 HashType defined by LUDefEntry 1 HashType defined via command
Direct_ HashType	4	Imm16(8..5)	GPR(3..0)	Use defined Hash Type for Hashing. Valid when DIrect_HashType_En = 1
Key	192	Shared Memory Pool		Provides pattern to be searched. Must be initialized before search. Located in the Shared Memory Pool.
KeyLen	8	Shared Memory Pool		Defines length of pattern minus 1 in key. Must be initialized before search. Located in the Shared Memory Pool.
Color	16	Shared Memory Pool		Must be initialized before search - used only when enabled in LUDefTable. Located in the Shared Memory Pool. Invalid when Direct_HashType_En is set to value '1'.

Table 183: Hash Key GTH Output Results

Result	Bit Length	Source	Description
HashedKeyReg	192	Register	Contains the HashedKey, including color when color is enabled in LUDefTable, according to section *8.2.1 Input Key and Color Register for FM and LPM Trees* on page 226 and section *8.2.2 Input Key and Color Register for SMT Trees* on page 227. Hash function is defined in LUDefTable. Hashed Key is NOT stored in Shared Memory Pool.
HashedKeyLen	8	Register	Contains length of pattern minus 1 in HashedKeyReg. Hashed Key is NOT stored in Shared Memory Pool.
DTA	26	Register	DTEntry Address (Hashed Key is NOT stored in Shared Memory Pool.)
LUDefCopy	96	Register	Set to contents of LUDefTable at entry given by LUDefIndex. Note: Valid for GTH Only.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.2 Read LUDefTable GTH (RDLUDEF GTH)

RDLUDEF reads the LUDefTable at a specified entry and stores the result in the LUDefCopy register. The TSE can read LUDefTable while picocode builds a key because *RDLUDEF* is executed asynchronously. Once the key is ready, the tree search execution can be executed with the UseLUDefCopyReg flag set to '1'.

Table 184: RDLUDEF_GTH Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable

Table 185: RDLUDEF_GTH Output Results

Result	Bit Length	Source	Description
LUDefCopy	96	Register	Scalar register contains content of LUDefTable at the entry. No content will be written back to Shared Memory Pool.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.3 Tree Search Enqueue Free List (TSENQFL)

TSENQFL releases a control block such as a leaf or PSCB to a free list. The address of the memory location to be freed is stored in LCBA0 / 1. The leaf or PSCB index to the free list is provided by the LUDefTable or directly by the command line. The enqueue operation always adds an address to the bottom of a free list FIFO-style. Entries cannot be added or removed from the middle of a free list. When FreeListCtrl =11, TSENQFL increments the LeafCount field in LUDefTable.

Table 186: TSENQFL Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
FreeListCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct Free List Index 10 PSCB Free List Index from LUDefTable 11 Leaf Free List Index from LUDefTable
LUDefIndex/ FreeListIndex	8	Imm16(12..5)	GPR(7..0)	Defines the entry in the LUDefTable used to read free list index information or directly defines FreeListIndex.
SrcType	3	Imm16(2..0)	Imm12(2..0)	000 LCBA0 001 LCBA1 100 PSCB0.Addr (for GCH only) 101 PSCB1.Addr (for GCH only) 110 PSCB2.Addr (for GCH only)
LCBA0 / 1 PSCB0 / 1 / 2.Addr	26	Register		The address to be freed or enqueued

Table 187: TSENQFL Output Results

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.4 Tree Search Dequeue Free List (TSDQFL)

TSDQFL dequeues an address from a given FQlinklist. The address that has been dequeued from the free list is stored in LCBA0 / 1. The leaf or PSCB index to the free list is provided by the LUDefTable or directly by the command line. When FreeListCtrl is '11', TSDQFL decrements the LeafCount field in LUDefTable.

Table 188: TSDQFL Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
FreeListCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct free list index 10 PSCB free list index from LUDefTable 11 Leaf free list index from LUDefTable
LUDefIndex/ FreeListIndex	8	Imm16(12..5)	GPR(7..0)	Defines the entry in LUDefTable used to read free list index information or directly defines FreeListIndex.
TgtType	3	Imm16(2..0)	Imm12(2..0)	000 LCBA0 001 LCBA1 100 PSCB0.Addr (for GCH only) 101 PSCB1.Addr (for GCH only) 110 PSCB2.Addr (for GCH only)

Table 189: TSDQFL Output Results

Result	Bit Length	Source	Description
LCBAO/1 PSCB0 / 1 / 2.Addr	26	Register	Dequeued address
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.5 Read Current Leaf from Rope (RCLR)

RCLR is used to “walk the rope” for such processes as aging. RCLR reads the leaf at the current leaf address defined in LUDefTable. It stores the leaf address in LCBA0 and the leaf contents in TSR0. When rope walking begins, the TSE invokes RCLR and picocode saves the leaf address (LCBA0) in a GPR. Before reading the next leaf, the TSE invokes the advance rope with optional delete leaf command (ARDL), after which RCLR can be invoked again. To determine whether a rope walk has ended, picocode compares LCBA0 with the GPR to verify whether the leaf that was read is the same as the first leaf. If the leaf is the same, the rope walk has ended.

At any time during the rope walk, picocode can delete a leaf from the rope using ARDL with the DeleteLeaf flag set to 1. This is useful when the leaf is to be aged out. RCLR can automatically delete leaves from the rope when the DeletePending bit in the leaf is set. When this feature is enabled, the TSE deletes a leaf and reads the next leaf, which is also deleted when the DeletePending bit is set. The process is repeated to delete multiple leaves.

After RCLR has executed with OK = 1, the contents of TSR0 and LCBA0 correspond with CLA in the LUDefTable, and the previous leaf on the rope has an address of PLA. OK = 0, or KO, means the rope is empty.

Table 190: RCLR Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable used to read rope
Reserved	3	Imm16(3..1)	Imm12(3..1)	Must be set to ‘000’
DelEnable	1	Imm16(0)	Imm12(0)	When ‘1’, leaves with DeletePending bit are automatically deleted from rope and leaf address will be enqueued in leaf free queue
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	Location for the leaf to be stored. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.

Table 191: RCLR Output Results

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
LCBA0	26	Register	Address of leaf that has been read
TSRx	512	Shared Memory Pool	Leaf content is stored in TSRx. TSRx is mapped into the Shared Memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA parameter for a length of 4 QW.

8.2.8.6 Advance Rope with Optional Delete Leaf (ARDL)

ARDL advances the rope, or updates CLA and PLA in LUDefTable. The NLA field from the leaf already stored in TSR0 is read and stored in CLA. PLA is then updated to the previous value of CLA unless the leaf is deleted. In this case, PLA remains the same and the NLARope field for the leaf with the current PLA address is set to the new CLA. When leaf deletion is enabled, the current leaf is deleted prior to advancing the rope. The contents of TSR0 and LCBA0 can be destroyed because ARDL uses TSR0 and LCBA0 as work areas to update the NLARope field. After ARDL is executed, CLA and PLA (in the LUDefTable) are updated and RCLR (described in the next section) can be executed again.

Table 192: ARDL Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable
DeleteLeaf	1	Imm16(0)	Imm12(0)	Enable deletion of current leaf from rope. 0 Do not delete leaf. 1 Delete leaf
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	Location of the current leaf address. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary.
TSRx	26	Register		Contents of current leaf (address CLA in LUDefTable). TSRx is mapped into the Shared Memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA parameter for a length of 4 QW.

Table 193: ARDL Output Results

Result	Bit Length	Source	Description
TSRx	512	Shared Memory Pool	Contents of TSRx will not be destroyed
LCBA0	26	Register	Contents of LCBA0 have been destroyed
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.7 Tree Leaf Insert Rope (TLIR)

TLIR inserts a leaf into the rope. The leaf must already be stored in TSR0 (done automatically by picocode during TLIR) and the leaf address must already be available in LCBA0. TLIR maintains the rope, which involves updating the PVA field in LUDefTable, the NLARope leaf field in Control Store, and the NLARope leaf field stored in TSRx. The leaf is inserted into the rope ahead of the current leaf, which has address CLA. Field PLA is updated to the new leaf address, which is LCBA0 / 1, in LUDefTable. Following TLIR execution, picocode must invoke MWR to write the leaf into Control Store. The contents of TSR1 and LCBA1 can be destroyed because TLIR uses TSR1 and LCBA1 as a work area.

Table 194: TLIR Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable
LCBA0	26	Register		Address of leaf to be inserted into rope
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Only values of x'8', x'A', x'C', and x'E' should be used.
TSRx	26	Shared Memory Pool		Contents of leaf to be inserted into rope. TSRx is mapped into the Shared Memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA parameter for a length of 4 QW.

Table 195: TLIR Output Results

Result	Bit Length	Source	Description
TSRx	512	Shared Memory Pool	Leaf NLA field has been updated
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.8 Clear PSCB (CLRPSCB)

This command writes all zeros to PSCB0 / 1 / 2.

Table 196: CLRPSCB Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
PN	2	Imm16(1..0)	--	Selects PSCB0, PSCB1, or PSCB2 register

Table 197: CLRPSCB Output Results

Result	Bit Length	Source	Description
PSCB<PN>	--	Register	Set to all zeros
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.9 Read PSCB (RDPSCB)

RDPSCB reads a PSCB from the Control Store and stores it in one of the PSCB0 / 1 / 2 registers. For LPM, the entire PSCB is read from memory at the address given by PSCB<pn>.Addr and stored in PSCB<pn>. For FM, the entire PSCB is read from memory and converted into a LPM PSCB format.

Table 198: RDPSCB Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
PN	2	Imm16(1..0)	Imm12(1..0)	Selects PSCB0, PSCB1, or PSCB2 register
PSCB<PN>.Addr	26	Register		Address in Control Store of PSCB to be read
RdWrPSCB_Cntl	2	Imm16(3..2)	Imm12(3..2)	00 Result is based on PatBit value. PatBit=0 Branch 0 entry of the PSCB is read PatBit=1 Branch 1 entry of the PSCB is read 01 Branch 0 entry of the PSCB is read 10 Branch 1 entry of the PSCB is read 11 Branch 0 and branch 1 entry of the PSCB is read
RdWrPSC_ DT_Flag	1	Imm16(4)	Imm12(4)	Indicates entry is DT

Table 199: RDPSCB Output Results

Result	Bit Length	Source	Description
PSCB<PN>	--	Register	PSCB is read from Control Store and stored in register PSCB<pn>. NPA0, NBT0, LCBA0 and NPA1, NBT1, LCBA1 fields are changed. Remainders are not changed.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.10 Write PSCB (WRPSCB)

WRPSCB writes a PSCB stored in PSCB0, 1, or 2 to the Control Store. For LPM, the entire PSCB is written to memory to the address given by PSCB<pn>.Addr. For FM, the PSCB is written to memory in FM PSCB format. An error flag is set when the PSCB is not in FM PSCB format. In both cases, WRPSCB generates the PSCB's two format bits. When the PSCB represents a DTEntry, only half of PSCB is written to memory. That is, the entire DTEntry is written.

Table 200: WRPSCB Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
PN	2	Imm16(1..0)	Imm12(1..0)	Selects PSCB0, PSCB1, or PSCB2 register
PSCB<PN>.Addr	26	Register		Address in Control Store of the PSCB to be written
RdWrPSCB_Cntl	2	Imm16(3..2)	Imm12(3..2)	00 Action is based on PatBit value. PatBit=0 Branch 0 entry of the PSCB is written PatBit=1 Branch 1 entry of the PSCB is written 01 Branch 0 entry of the PSCB is written 10 Branch 1 entry of the PSCB is written 11 Branch 0 and branch 1 entry of the PSCB is written
RdWrPSC_DT_Flag	1	Imm16(4)	Imm12(4)	Indicates entry is DT

8.2.8.11 Push PSCB (PUSHPSCB)

PUSHPSCB pushes the PSCB stack.

Table 201: PUSHPSCB Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
PSCB0	--	--		Contains a PSCB
PSCB1	--	--		Contains a PSCB

Table 202: PUSHPSCB Output Results

Result	Bit Length	Source	Description
PSCB2	--	Register	Set to PSCB1
PSCB1	--	Register	Set to PSCB0
PSCB0	--	Register	Set to all zeros
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.12 Distinguish (DISTPOS)

DISTPOS performs a pattern compare between the patterns stored in HashedKey and TSR0. The result is stored in the DistPosReg register. The OK flag is set when a full match has been detected.

Table 203: DISTPOS Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
HashedKey	192	Register		Contains hashed pattern
HashedKeyLen	8	Register		Contains length of hashed pattern minus 1
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.
TSRx	512	Shared Memory Pool		Contains second pattern with pattern length (for LPM only). TSRx is mapped into the Shared Memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA parameter for a length of 4 QW.

Table 204: DISTPOS Output Results

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 Pattern does not match 1 Pattern in HashedKey matches pattern in TSRx
DistPos	8	Register	Smallest bit position where pattern in HashedKey differs from pattern in TSRx
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.13 TSR0 Pattern (TSR0PAT)

TSR0PAT reads a bit from the pattern stored in TSRx and stores this bit in the PatBit_TSR0 register.

Table 205: TSR0PAT Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
BitNum	8	--	GPR(7..0)	Selects bit in TSR0 pattern
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.

Table 206: TSR0PAT Output Results

Result	Bit Length	Source	Description
PatBit_TSR0	1	Register	Set to value of bit BitNum of pattern stored in TSR0
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.8.14 Pattern 2DTA (PAT2DTA)

PAT2DTA reads a pattern from TSRx, stores it in the HashedKey, and sets the DTA register accordingly. PAT2DTA does not perform a hash since the pattern in a leaf is already hashed. Pattern read from TSRx is assumed to be already hashed.

Table 207: PAT2DTA Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
LUDefIndex	8	Imm16(12..5)	GPR(7..0)	Defines entry in LUDefTable used to calculate DTA from HashedKey
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.10 Shared Memory Pool* on page 198) and is constrained to be on a four-QW boundary. Use only values of x'8', X'A', X'C', and X'E'.

Table 208: PAT2DTA Output Results

Result	Bit Length	Source	Description
DTA	26	Register	DTEntry Address corresponding to DT definition in LUDefTable and HashedKey
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.9 Hash Functions

Table 209: General Hash Functions

Hash_type	Name	Description
0	No hash	No hash is performed and hashed output H(191..0) equals input key K(191..0). Can be used for SMT trees or LPM trees if 32-bit IP LPM hash cannot be used.
1	192-bit IP Hash	Uses four copies of IP hash box. See *Figure 61: 192-Bit IP Hash Function* on page 257
2	192-bit MAC Hash	See *Figure 62: MAC Hash Function* on page 258
3	192-bit Network DISTPOS	See *Figure 63: Network Dispatcher Hash Function* on page 259
4	Reserved	
5	48-bit MAC swap	See *Figure 64: 48-Bit MAC Hash Function* on page 260
6	60-bit MAC swap	See *Figure 65: 60-Bit MAC Hash Function* on page 261
7	192-bit SMT key	See *Figure 60: No-Hash Function* on page 256
8	Reserved	
9	Reserved	
10	Reserved	

In the following figures, the input is always the 192-bit key and the output is a 192-bit hashed output before color insertion. If color is enabled, the color is inserted at the bit position given by DTSize in the LUDefTable and 16 LSBs of the key are ignored since maximum key length of 176 bits is supported when color is enabled.

Figure 60: No-Hash Function

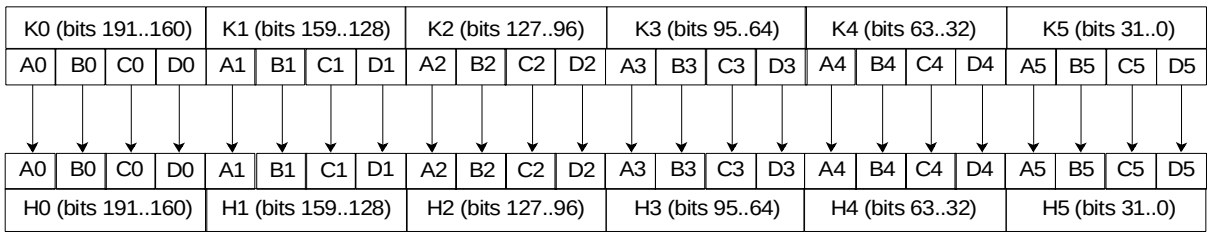


Figure 61: 192-Bit IP Hash Function

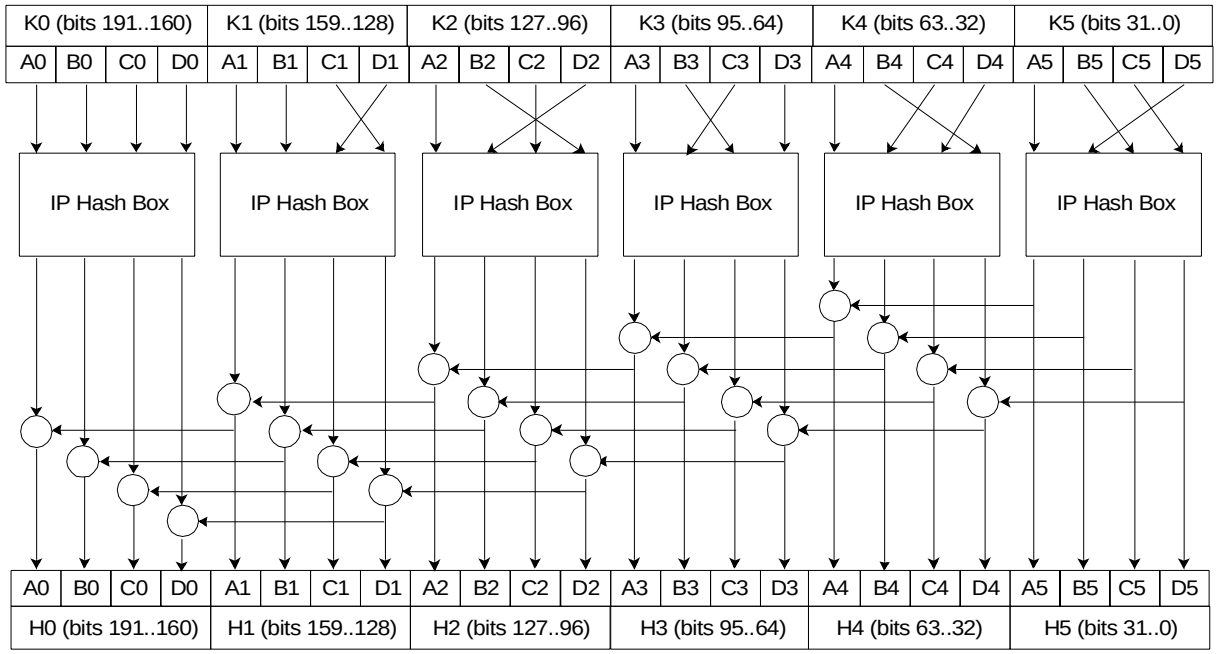


Figure 62: MAC Hash Function

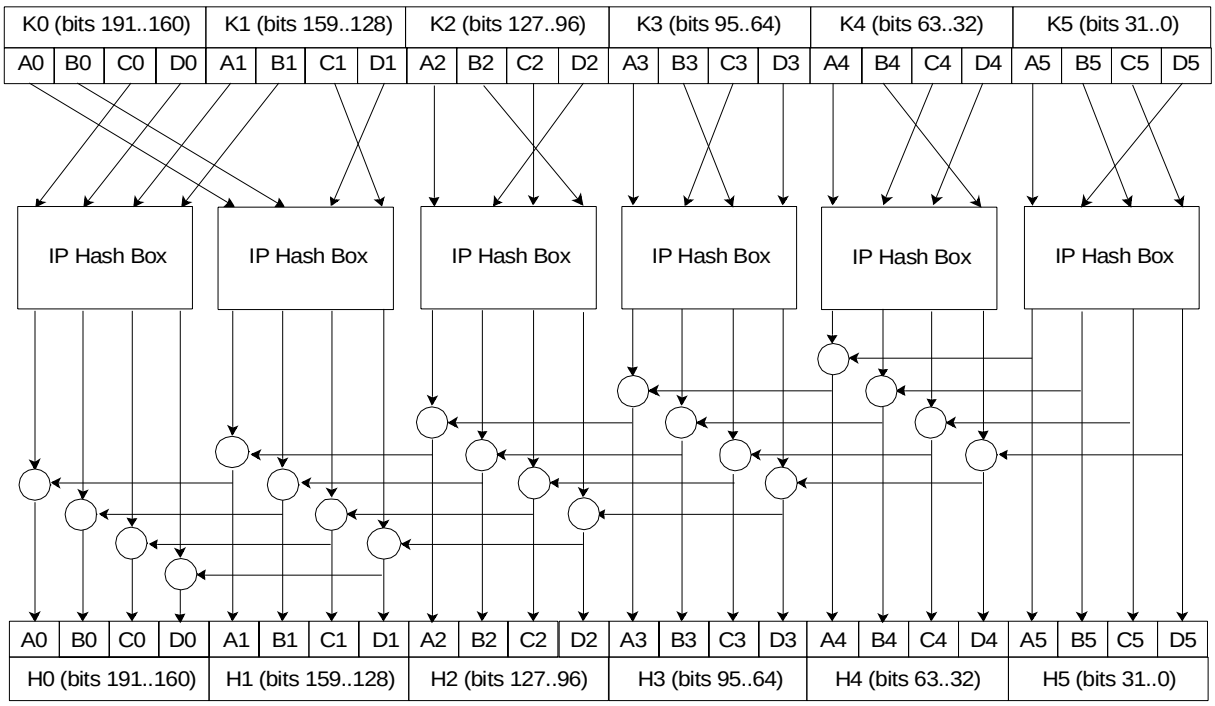


Figure 63: Network Dispatcher Hash Function

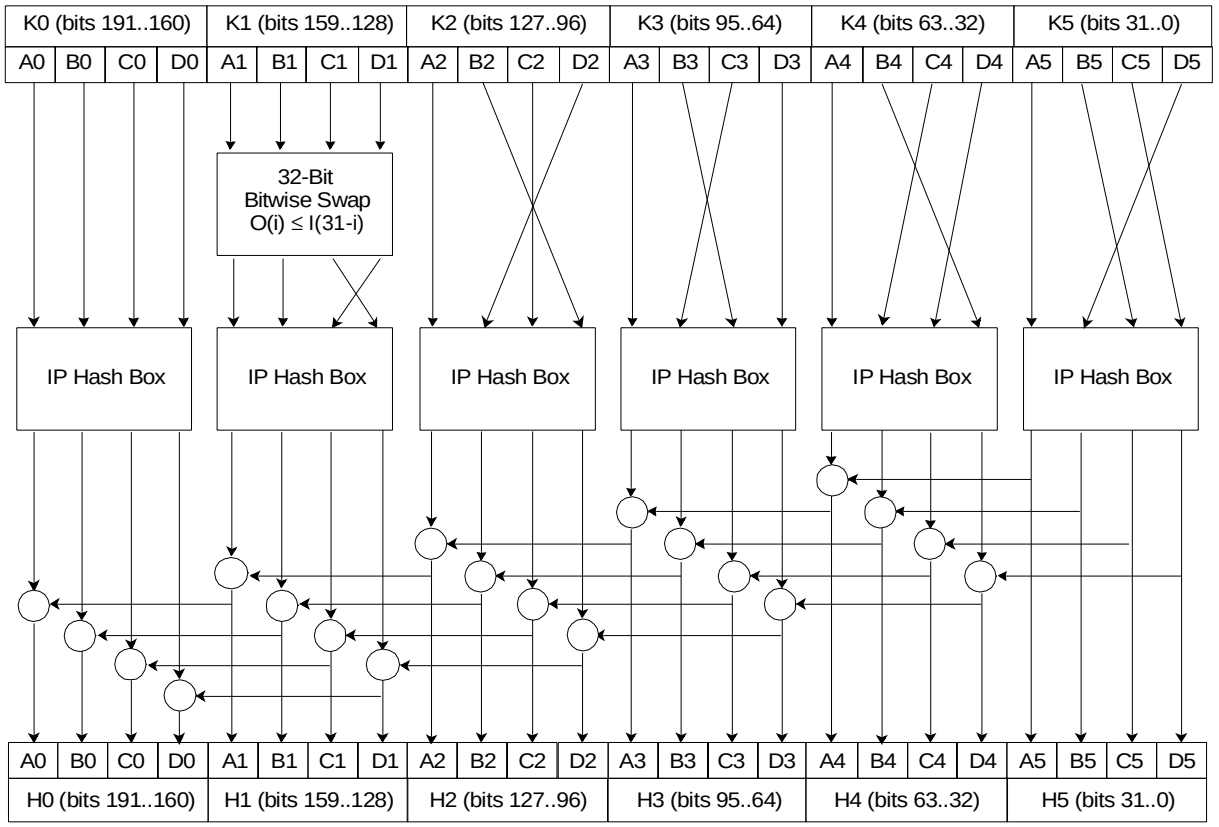


Figure 64: 48-Bit MAC Hash Function

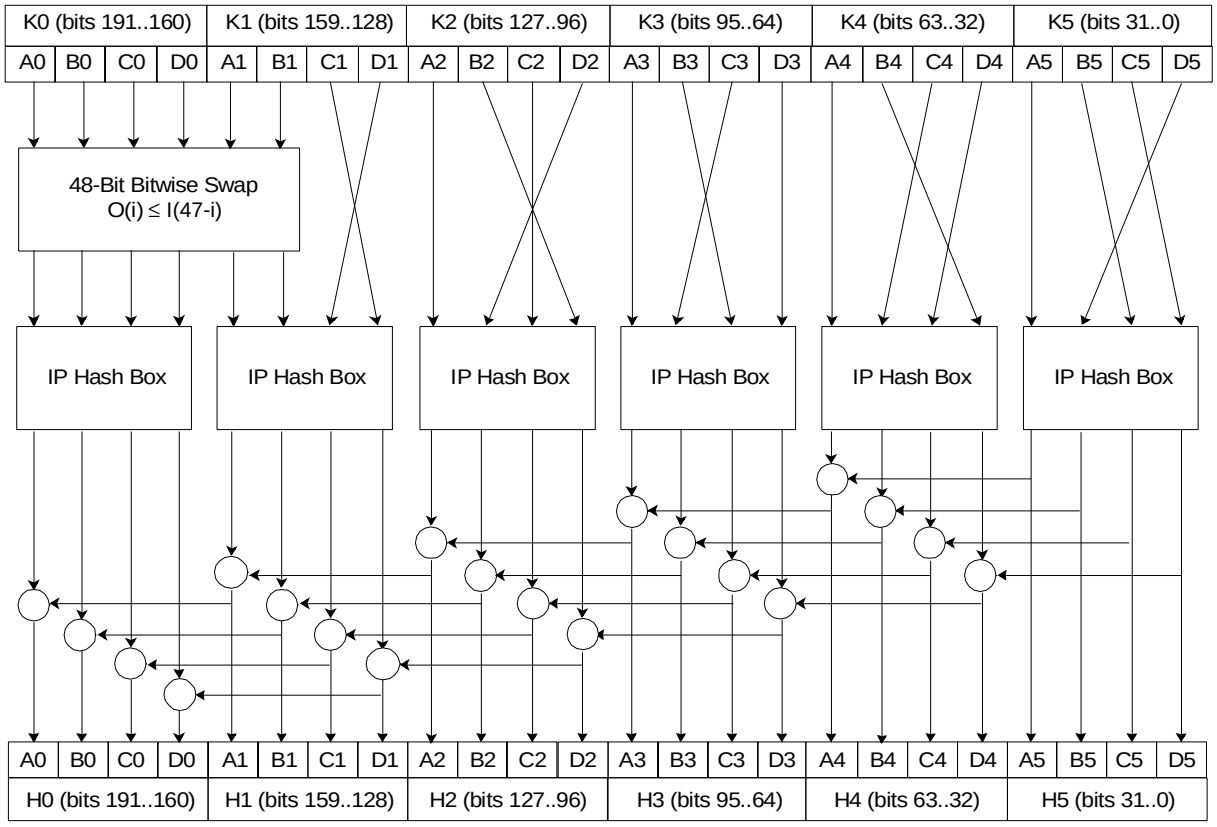
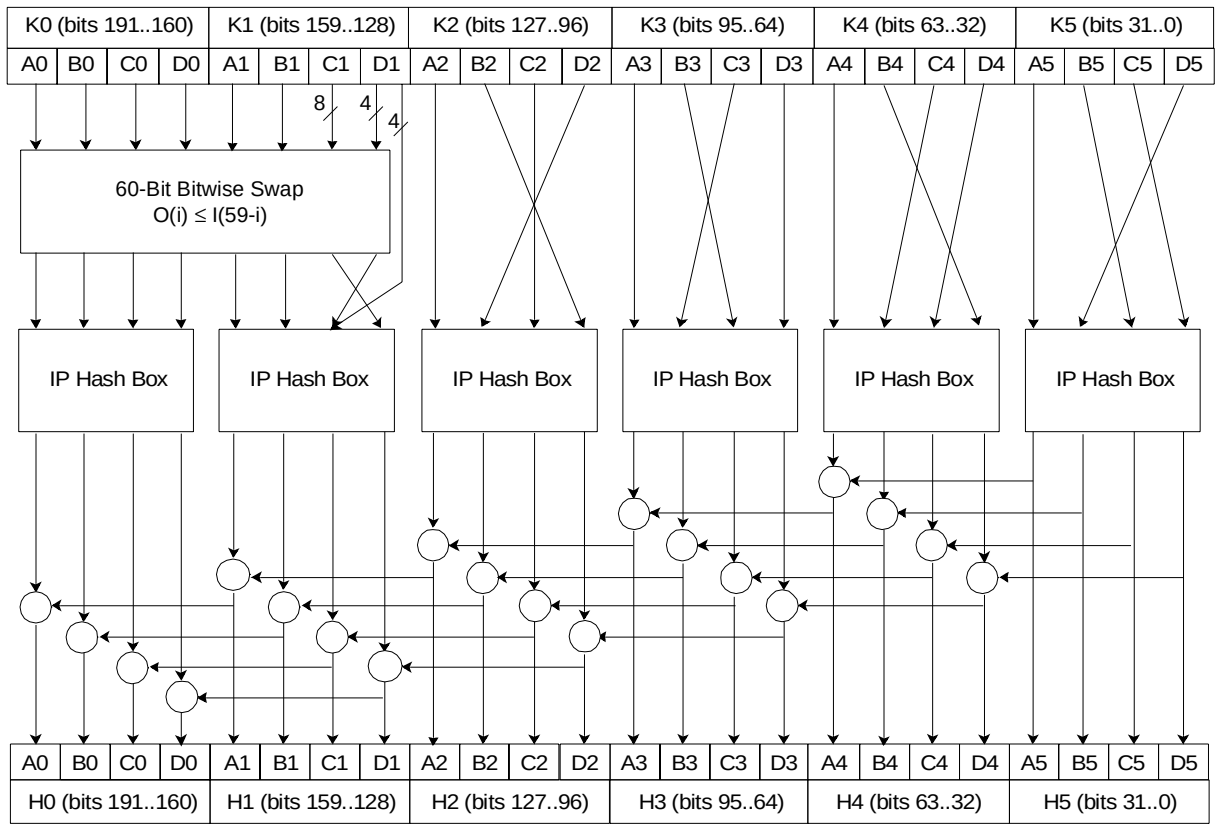


Figure 65: 60-Bit MAC Hash Function



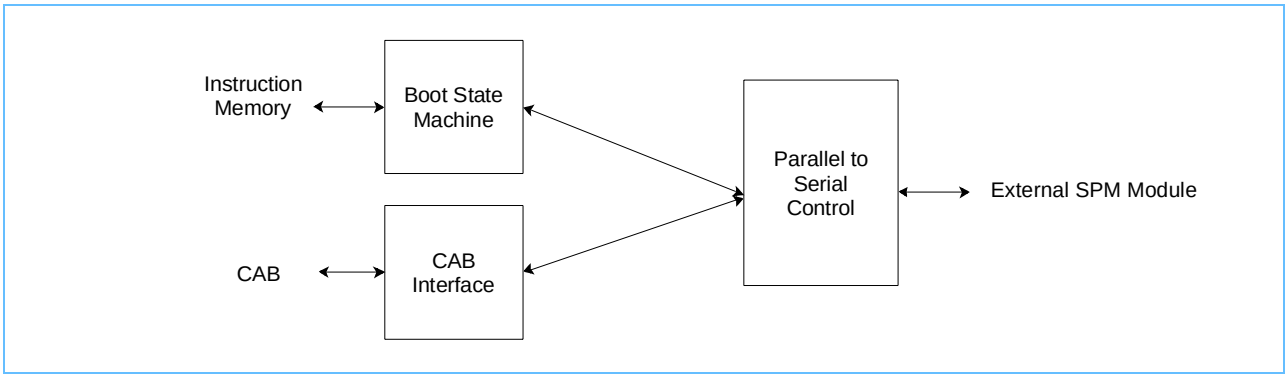
9. Serial/Parallel Manager Interface

The Serial/Parallel Manager (SPM) Interface is a serial interface to communicate with external devices. The SPM Interface consists of a clock signal output, a bi-directional data signal, and an interrupt input. On this interface, the NP4GS3 is the master and the external SPM module is the only slave[1]. The SPM Interface loads picocode, allowing management of physical layer components and access to card-based functions such as LEDs.

The SPM Interface supports:

- An external SPM module
- Boot code load via external SPM and EEPROM
- Boot override via CABWatch interface or Boot_Picocode configuration chip I/O.
- Access to external PHY, LED, management, and card-based functions

Figure 66: SPM Interface Block Diagram



9.1 SPM Interface Components

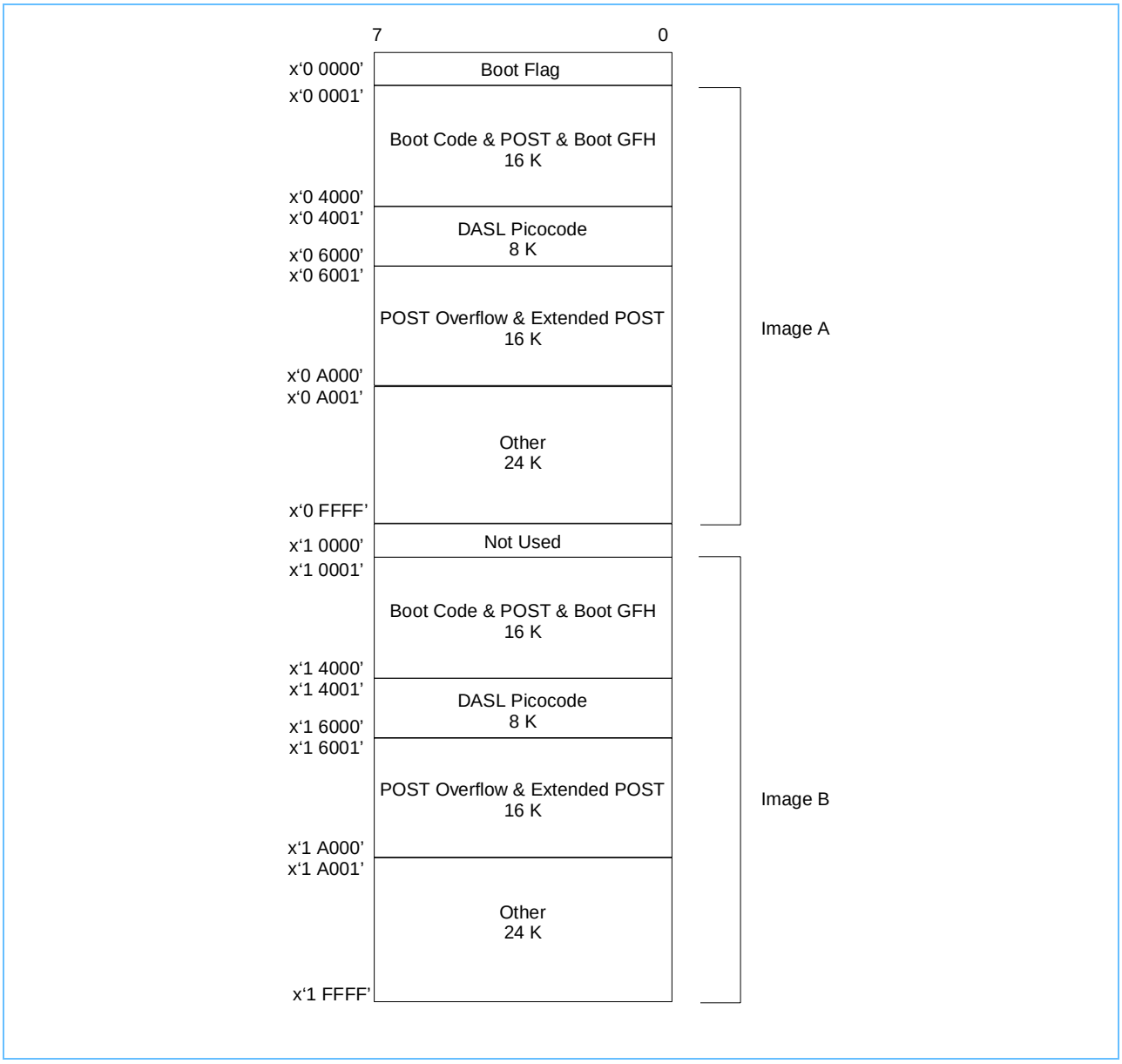
Boot State Machine	Starts up after reset if configured to do so by an external IO pin (Boot_Picocode set to 0). It selects one of two boot images (picocode loads) based on a configuration flag found in the EEPROM and places the code into the Instruction memory in the EPC. Once the code is loaded, the Boot State Machine causes an interrupt that starts up the GFH. The GFH executes the loaded code.
CAB Interface	A memory mapped interface to the NP4GS3 that allows the protocol processors to access any external device, including an SPM module, external PHYs, or card LEDs.
Parallel to Serial Control	Converts between the internal 32-bit read/write parallel interface and the 3-bit external bi-directional serial interface.

1.The external SPM module is not supplied by IBM.

9.2 SPM Interface Data Flow

The SPM Interface is used initially to boot the NP4GS3. Following a reset, the SPM Interface reads an external EEPROM and loads the EEPROM's contents into the EPC's Instruction Memory. When loading is complete, the SPM Interface issues a "boot done" interrupt which causes the Guided Frame Handler (GFH) to start executing the boot code. The boot code initializes the network processor's internal structures and configures all the interfaces that the network processor requires to operate. When all boot processing is complete, the GFH activates the operational signal to indicate its availability to the Control Point Function (CPF). The CPF sends guided frames to further initialize and configure the network processor, preparing it for network operation.

Figure 67: EPC Boot Image in External EEPROM



The Boot State Machine supports two images of boot code in external EEPROM. The contents of byte x'0 0000' in EEPROM is examined during the read process to determine which image to load. When the most significant bit of this byte is a '0', the current image resides at addresses x'0 0001' - x'0 4000'. Otherwise, the current image resides at addresses x'1 0001' - x'1 4000'. The Boot State Machine will load the appropriate image and allow the other image area to be used for boot code updates.

The SPM Interface is also used during initialization and statistics gathering. It interfaces to the Ethernet PHYs, card LEDs, and other card-level structures through an external SPM interface module supplied by the customer. These external structures are mapped to the CAB address space and are accessed from the picocode using the same methods as those used to access any internal data structures: the picocode issues reads or writes to the appropriate address and the SPM Interface converts these reads and writes to serial communications with the external SPM module. The external SPM module re-converts these serial communications to register reads and writes and to access the desired card device. Through the SPM interface, the picocode has indirect access to all card-level functions and can configure or gather statistics from these devices as if they were directly attached to the CAB interface.

9.3 SPM Interface Protocol

The SPM Interface operates synchronously with respect to the 33 MHz clock signal output. Data, address, and control information is transferred serially on a bidirectional data signal. Transitions on this data signal occur at the rising edges of the clock signal. Each data exchange is initiated by the NP4GS3 and can be one to four bytes in length.

For single byte transfers, the exchange begins when the NP4GS3 drives the data signal to '1' for one clock period. This "select" indication is followed by a 1-bit write/$\overline{\text{read}}$ indication, a 4-bit burst length indication, and a 25-bit address value.

For read and write transfers, when the SDM interface master is waiting for a response from the SDM interface slave, the slave communicates with the master using the following:

- ack: a '1' driven onto the databus by the SDM interface slave to indicate each successful byte operation, either read or write.
- $\overline{\text{ack}}$: a '0' driven onto the databus by the SDM interface slave while the byte operation is in progress.

For write transfers (see *Figure 70: SPM Interface Write Protocol* on page 266), the address is followed immediately by eight bits of data. The NP4GS3 then puts its driver in High-Z mode. One clock period later, the slave drives the data signal to '0' ($\overline{\text{ack}}$) until the byte write operation is complete. The slave then drives the data signal to '1' (ack). During the byte write operation, the NP4GS3 samples the data input looking for a '1' (ack). Immediately following the ack, the slave puts its driver in High-Z mode. The transfer is concluded one clock period later.

For read transfers (see *Figure 69: SPM Interface Read Protocol* on page 266), the address is followed by the NP4GS3 putting its driver into High-Z mode. One clock period later, the slave drives the data signal to '0' ($\overline{\text{ack}}$) until the byte of read data is ready for transfer. The slave then drives the data signal to '1' (ack). During this time, the NP4GS3 samples the data input looking for an ack. Immediately following the ack, the slave drives the eight bits of data onto the data signal and then puts its driver in High-Z mode. The transfer is concluded one clock period later.

The protocol for multiple byte transfers is similar except that each byte written is accompanied by a bus turnaround, zero or more $\overline{\text{acks}}$, and an ack. Read bursts are characterized by the slave retaining bus ownership until the last byte of the burst is transferred. Each successive byte read is preceded by at least one '0' on the

databus followed by one ‘1’ on the databus (ack), and immediately followed by the eight bits of data. Examples of the Read and Write protocols are shown in *Figure 69* and *Figure 70* respectively.

Figure 68: SPM Bit Timing

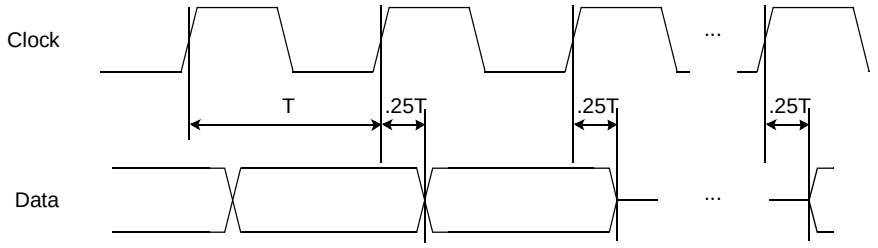


Figure 69: SPM Interface Read Protocol

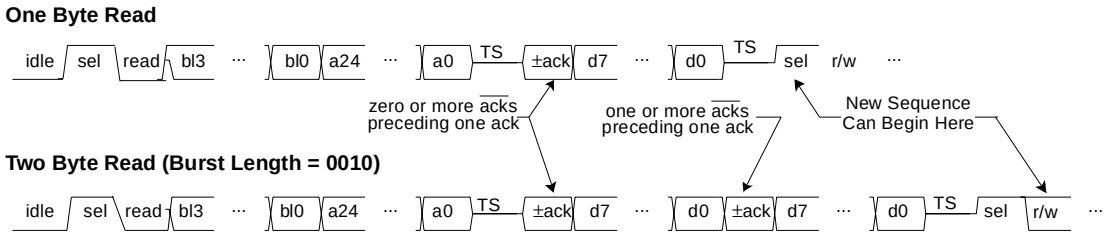
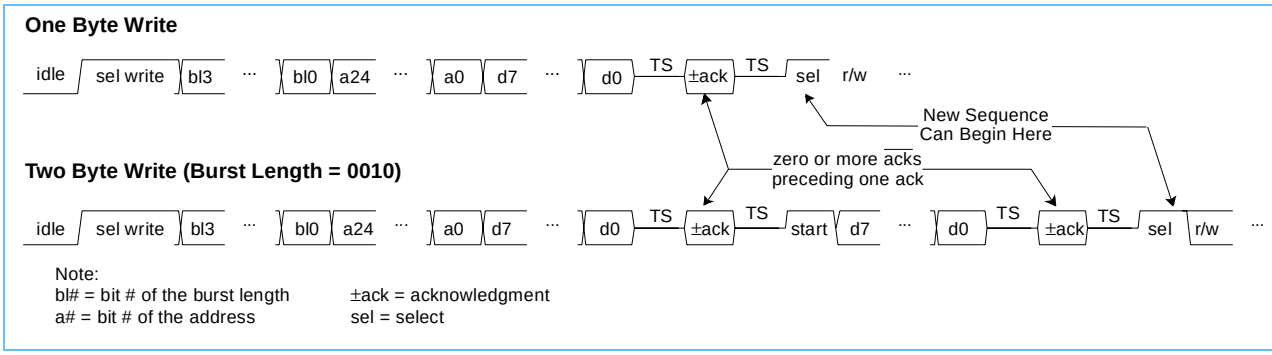


Figure 70: SPM Interface Write Protocol



9.4 SPM CAB Address Space

The SPM interface provides CAB access to an external EEPROM and to other devices attached via an external SPM module developed by the customer. The address space is broken into three areas:

- Byte access
- Word access
- EEPROM access

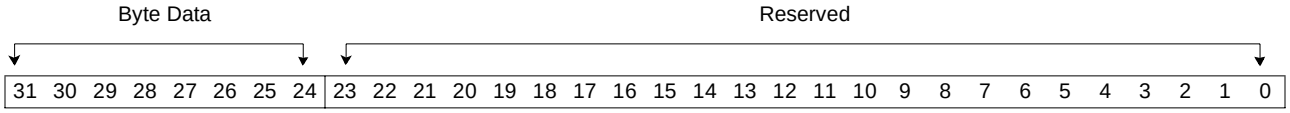
9.4.1 Byte Access Space

All elements accessed in byte access space are limited to a single byte in width.

Access Type R/W

Base Addresses x‘2800 0000’ through x‘281F FFFF’

x‘2880 0000’ through x‘28FF FFFF’



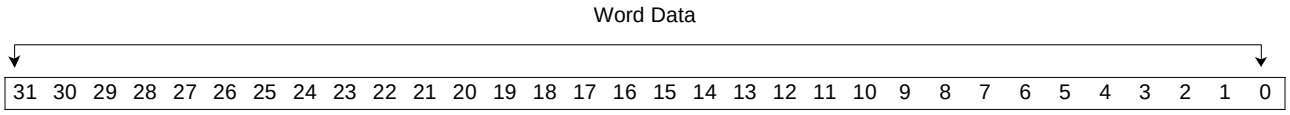
Field Name	Bit(s)	Reset	Description
Byte Data	31:24		Data at this location
Reserved	23:0		Reserved

9.4.2 Word Access Space

All elements accessed in word access space are a word in width.

Access Type: R/W

Base Addresses: x‘2820 0000’ through x‘287F FFFF’



Field Name	Bit(s)	Reset	Description
Word Data	31:0		Data at this location

9.4.3 EEPROM Access Space

The SPM interface and a customer supplied SPM module can be combined to provide access to an attached EEPROM. The EEPROM access space contains locations for 8224 1-byte elements. All write accesses are limited to a single byte, but read accesses may be in bursts of 1, 2, 3, or 4 bytes. The CAB address is formed using the field definitions shown in *Table 210*.

Table 210: Field Definitions for CAB Addresses

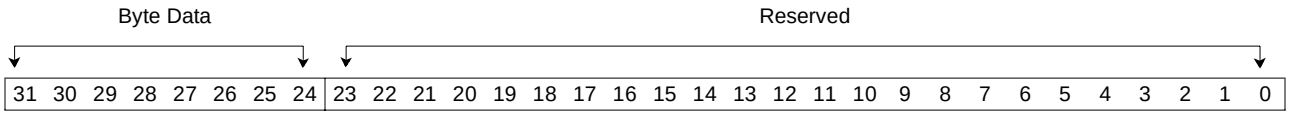
Bits	Description
31:27	'00101'
26:25	Encoded burst length 00 4-byte burst (read only) 01 1-byte burst (read or write) 10 2-byte burst (read only) 11 3-byte burst (read only)
24	'1'
23:0	Starting Byte address for read or write action

9.4.3.1 EEPROM Byte Access

Addresses in this space are used for single-byte read or write access to the EEPROM.

Access Type: R/W

Base Addresses: x‘2B00 0000’ through x‘2BFF FFFF’



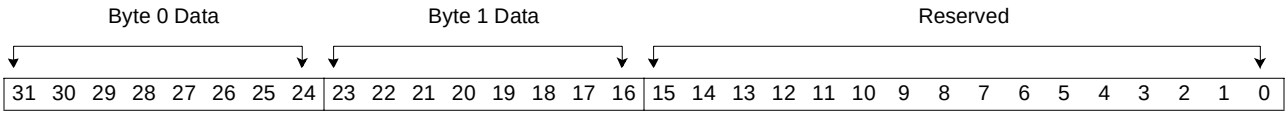
Field Name	Bit(s)	Reset	Description
Byte Data	31:24		Data at starting byte address
Reserved	23:0		Reserved

9.4.3.2 EEPROM 2 Byte Access

Addresses in this space are used for a 2-byte read burst access to the EEPROM.

Access Type: Read Only

Base Addresses: x‘2D00 0000’ through x‘2DFF FFFF’



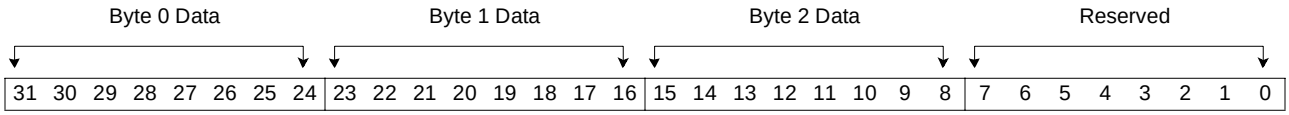
Field Name	Bit(s)	Reset	Description
Byte 0 Data	31:24		Data at starting byte address
Byte 1 Data	23:16		Data at starting byte address + 1
Reserved	15:0		Reserved

9.4.3.3 EEPROM 3 Byte Access

Addresses in this space are used for a 3-byte read burst access to the EEPROM.

Access Type: Read Only

Base Addresses: x‘2F00 0000’ through x‘2FFF FFFF’



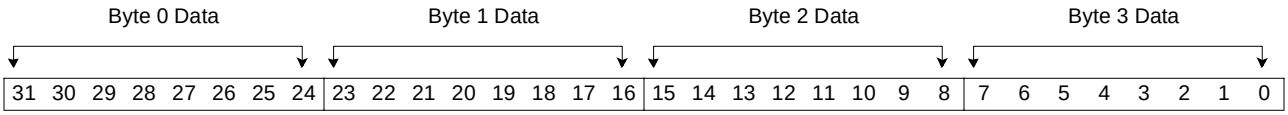
Field Name	Bit(s)	Reset	Description
Byte 0 Data	31:24		Data at byte address
Byte 1 Data	23:16		Data at byte address + 1
Byte 2 Data	15:8		Data byte address + 2
Reserved	7:0		Reserved

9.4.3.4 EEPROM 4 Byte Access

Addresses in this space are used for a 4-byte read burst access to the EEPROM.

Access Type: Read only

Base Addresses: x‘2900 0000’ through x‘29FF FFFF’



Field Name	Bit(s)	Reset	Description
Byte 0 Data	31:24		Data at byte address
Byte 1 Data	23:16		Data at byte address + 1
Byte 2 Data	15:8		Data byte address + 2
Byte 3 Data	7:0		Data byte address + 3

10. Embedded PowerPC™

10.1 Description

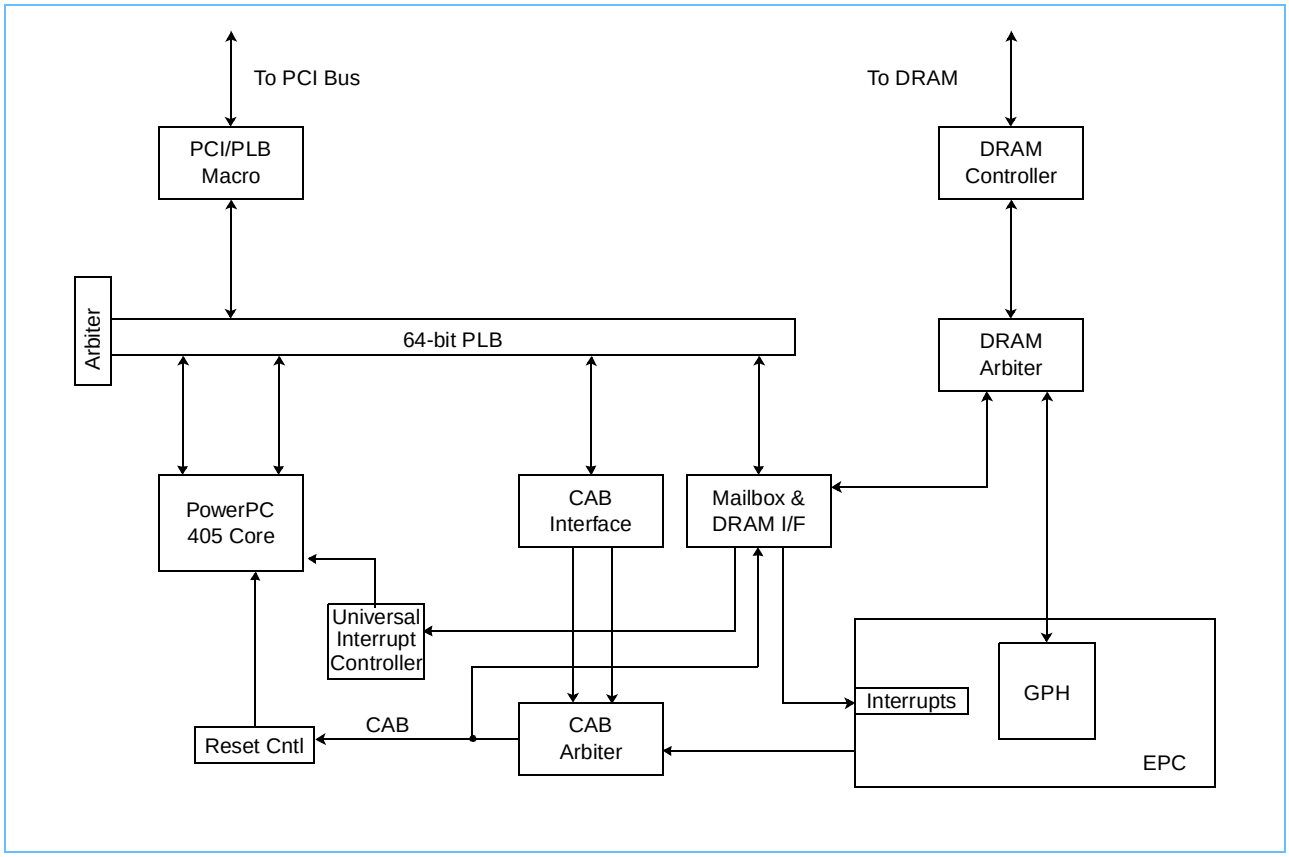
The NP4GS3 incorporates an embedded PowerPC subsystem. This subsystem consists of mixture of macros from IBM's PowerPC macro library and other components that were designed specifically for the NP4GS3.

Standard IBM PowerPC macros include the following:

- 133 MHz 405 Processor Core with 16 K of instruction cache and 16 K of data cache
- 133 MHz, 64-bit PLB macro with PLB arbiter
- 33/66 MHz, 32-bit PCI to 133 MHz, 64-bit PLB macro
- PowerPC Universal Interrupt Controller (UIC) macro

The embedded PowerPC subsystem includes a CAB Interface PLB slave unit to access NP4GS3 internal structures and a Mailbox and DRAM Interface PLB slave unit for inter-processor communications and access to PowerPC instructions.

Figure 71: PowerPC Block Diagram



10.2 Processor Local Bus and Device Control Register Buses

The on-chip bus structure consisting of the Processor Local Bus (PLB) and the Device Control Register bus (DCR) provides a link between the processor core and the other peripherals (PLB master and slave devices) used in PowerPC subsystem design.

The PLB is the high performance bus used to access memory, PCI devices, and Network Processor structures through the PLB interface units. The PLB interface units shown in *Figure 71*, the CAB Interface and the Mailbox & DRAM interface, are PLB slaves. The processor core has two PLB master connections, one for instruction cache and one for data cache. The PCI to PLB interface unit, which is both a PLB master and PLB slave device, is also attached to the PLB. The PLB master corresponds to the PCI target and the PLB slave corresponds to the PCI master.

Each PLB master is attached to the PLB through separate address, read data, and write data buses and a plurality of transfer qualifier signals. PLB slaves are attached to the PLB through shared, but decoupled, address, read data, and write data buses and a plurality of transfer control and status signals for each data bus.

Access to the PLB is granted through a central arbitration mechanism that allows masters to compete for bus ownership. This mechanism is flexible enough to provide for the implementation of various priority schemes. Additionally, an arbitration locking mechanism is provided to support master-driven atomic operations.

The PLB is a fully-synchronous bus. Timing for all PLB signals is provided by a single clock source that is shared by all masters and slaves attached to the PLB.

Table 211: PLB Master Connections

Master ID	Master Unit Description
0	Processor Core Instruction Cache Unit
1	Processor Core Data Cache Unit
2	PLB/PCI Macro Unit
Others	Unused

All PLB arbiter registers are device control registers. They are accessed by using the Move From Device Control Register (mfdcr) and Move To Device Control Register (mtdcr) instructions. PLB arbiter registers are architected as 32-bits and are privileged for both read and write. The DCR base address of the PLB registers is x'080'. The DCR base address of the UIC registers is x'0C0'. Details regarding the PLB and UIC device control registers can be found in the PPC405GP Embedded Controller User's Manual (http://www.chips.ibm.com/products/powerpc/chips/).

The Device Control Register (DCR) bus is used primarily to access status and control registers within the PLB and the Universal Interrupt Controller (UIC). The DCR bus architecture allows data transfers among peripherals to occur independently from, and concurrent with, data transfers between the processor and memory or among other PLB devices.

10.3 Universal Interrupt Controller (UIC)

The Universal Interrupt Controller (UIC) provides all the necessary control, status, and communication between the various of interrupts sources and the microprocessor core. The UIC supports six on-chip and two external sources of interrupts. Status reporting (using the UIC Status Register (UICSR)) is provided to ensure that systems software can determine the current and interrupting state of the system and respond appropriately. Software can generate interrupts to simplify software development and for diagnostics.

The interrupts can be programmed, using the UIC Critical Register (UICCR), to generate either a critical or a non-critical interrupt signal.

The UIC supports internal and external interrupt sources as defined in Table 212.

Table 212: UIC Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Edge	DRAM D6 Parity Error
1	Programmable	Programmable	External Interrupt 0
2	Programmable	Programmable	External Interrupt 1
3	High	Level	PCI Host to PowerPC Doorbell Interrupt
4	High	Level	PCI Host to PowerPC Message Interrupt
5	High	Level	Embedded Processor Complex to PowerPC Doorbell Interrupt
6	High	Level	Embedded Processor Complex to PowerPC Message Interrupt
7	High	Level	PCI Command Write Interrupt
8-31			unused, interrupt input to UIC tied low.

The on-chip interrupts (interrupts 0, and 3-7) and the external interrupts (interrupts 1-2) are programmable. However, the on-chip interrupts must be programmed as shown in Table 212. For details regarding the control of the UIC, including the programming of interrupts, see the PPC405GP Embedded Controller User's Manual (http://www.chips.ibm.com/products/powerpc/chips/).

10.4 PCI/PLB Macro

The Peripheral Component Interconnect (PCI) interface controller provides an interface for connecting PLB-compliant devices to PCI devices. The controller complies with PCI Specification, version 2.2. (http://www.pcisig.com).

The PCI/PLB macro responds as a target on the PLB bus in several address ranges. These ranges allow a PLB master to configure the PCI/PLB macro, and to cause the PCI/PLB macro to generate memory, I/O, configuration, interrupt acknowledge, and special cycles to the PCI bus. Table 213 shows the address map from the view of the PLB, that is, as decoded by the PCI/PLB macro as a PLB slave.

Table 213: PLB Address Map for PCI/PLB Macro

PLB Address Range	Description	PCI Address Range
x‘E800 0000’ - x‘E800 FFFF’	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 0 to 64 KB - 1	x‘0000 0000’ - x‘0000 FFFF’
x‘E801 0000 x‘E87F FFFF‘	Reserved PCI/PLB Macro does not respond (Other bridges use this space for non-contiguous I/O).	
x‘E880 0000’ - x‘EBFF FFFF’	PCI I/O Accesses to this range are translated to an I/O access on PCI in the range 8 MB to 64 MB - 1	x‘0080 0000’ - x‘03FF FFFF’
x‘EC00 0000’ - x‘EEBF FFFF’	Reserved PCI Macro does not respond	
x‘EEC0 0000’ - x‘EECF FFFF’	PCICFGADR and PCICFGDATA x‘EEC0 0000’: CONFIG_ADDRESS x‘EEC0 0004’: CONFIG_DATA x‘EEC0 0008’ - x‘EECF FFFF’: Reserved (can mirror PCICFGADR and PCICFGDATA)	
x‘EED0 0000’ - x‘EEDF FFFF‘	PCI Interrupt Acknowledge and Special Cycle x‘EED0 0000’ read: Interrupt Acknowledge x‘EED0 0000’ write: Special Cycle x‘EED0 0004’ - x‘EEDF FFFF’: Reserved (can mirror Interrupt acknowledge and Special Cycle)	
x‘EEE0 0000’ - x‘EF3F FFFF’	Reserved PCI/PLB Macro does not respond	

Table 213: PLB Address Map for PCI/PLB Macro

PLB Address Range	Description	PCI Address Range
x‘EF40 0000’ - x‘EF4F FFFF’	PCI/PLB Macro Local Configuration Registers x‘EF40 0000’: PMM0LA x‘EF40 0004’: PMM0MA x‘EF40 0008’: PMM0PCILA x‘EF40 000C’: PMM0PCIHA x‘EF40 0010’: PMM1LA x‘EF40 0014’: PMM1MA x‘EF40 0018’: PMM1PCILA x‘EF40 001C’: PMM1PCIHA x‘EF40 0020’: PMM2LA x‘EF40 0024’: PMM2MA x‘EF40 0028’: PMM2PCILA x‘EF40 002C’: PMM2PCIHA x‘EF40 0030’: PTM1MS x‘EF40 0034’: PTM1LA x‘EF40 0038’: PTM1MS x‘EF40 003C’: PTM1MS x‘F400 0400’ - x‘EF4F FFFF’: Reserved (can mirror PCI local registers)	
x‘0000 0000’ - x‘FFFF FFFF’	PCI Memory - Range 0 PMM 0 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	x‘0000 0000 0000 0000’ x‘FFFF FFFF FFFF FFFF’
x‘0000 0000’ - x‘FFFF FFFF’	PCI Memory - Range 1 PMM 1 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	x‘0000 0000 0000 0000’ x‘FFFF FFFF FFFF FFFF’
x‘0000 0000’ - x‘FFFF FFFF’	PCI Memory - Range 2 PMM 2 registers map a region in PLB space to a region in PCI memory space. The address ranges are fully programmable. The PCI address is 64 bits.	x‘0000 0000 0000 0000’ x‘FFFF FFFF FFFF FFFF’

Following a general reset of the Network Processor, the PCI Target Map 1 is enabled for a PCI address range of 128 KB and is mapped to the PLB address range of x’7800 0000 - 7801 FFFF’. The corresponding PCI base address for this range must be set by PCI configuration of the PCI PTM1 Base Address Register. Likewise, the PCI Target Map 2 is enabled for a PCI address range of 128 MB and is mapped to the PLB address range of x’0000 0000 - 03FF FFFF’. The corresponding PCI base address for this range must be set by PCI configuration of the PCI PTM2 Base Address Register.

The PCI/PLB macro has a mode that enables a PLB master to access a PCI memory range without initial configuration cycles. This mode is enabled by strapping the boot_ppc input pin high. System designers, for instance, may use this mode to allow a processor to access a boot ROM in PCI memory space. in this mode the PCI/PLB macro comes out of reset with PMM0 enabled and programmed for the address range x’FFFE 0000’ - x’FFFF FFFF’. The ME field of the PCI Command register (PCICMD[ME]) is also set to 1 after reset. Enabling PCI boot mode does not prevent subsequent updates to the PMM0 registers.

The general reset initializes the PCI/PLB macro’s Bridge Options 2 Register (PCIBRDGOPT2) with its Host Configuration Enable bit set to ‘1’ (enabled). This allows an external source to access the PCI/PLB macro’s configuration registers.

For further details regarding the PCI/PLB macro’s control and configuration registers, see the PPC405GP Embedded Controller User’s Manual (http://www.chips.ibm.com/products/powerpc/chips/).

10.5 PLB Address Map

Components of the embedded PowerPC are connected using the Processor Local Bus (PLB). These components recognize PLB address values as their own. These PLB address values are fixed by hardware. The PLB address map describes the association of PLB address values and the components that recognize them.

Symbolic Address	PLB Address	Description	Access
CAB Interface Macro			
PwrPC_CAB_Addr	x'7800 0000'	PowerPC CAB Address Register	R/W
PwrPC_CAB_Data	x'7800 0008'	PowerPC CAB Data Register	R/W
PwrPC_CAB_Cntl	x'7800 0010'	PowerPC CAB Control Register	R/W
PwrPC_CAB_Status	x'7800 0018'	PowerPC CAB Status Register	R/W
Host_CAB_Addr	x'7800 8000'	PCI Host CAB Address Register	R/W
Host_CAB_Data	x'7800 8008'	PCI Host CAB Data Register	R/W
Host_CAB_Cntl	x'7800 8010'	PCI Host CAB Control Register	R/W
Host_CAB_Status	x'7800 8018'	PCI Host CAB Status Register	R/W
Unassigned addresses in the range x'7800 0000' - x'7800 FFFF' are reserved			
Mailbox and DRAM Interface Macro			
PCI_Interr_Status	x'7801 0000'	PCI Interrupt Status Register	R
PCI_Interr_Ena	x'7801 0008'	PCI Interrupt Enable Register	R/W
P2H_Msg_Resource	x'7801 0010'	PowerPC to PCI Host Resource Register	R&RS/W
P2H_Msg_Addr	x'7801 0018'	PowerPC to PCI Host Message Address Register	R/W
P2H_Doorbell	x'7801 0020'	PowerPC to PCI Host Doorbell Register (PowerPC Access)	R/SUM
	x'7801 0028'	PowerPC to PCI Host Doorbell Register (PCI Host Access)	R/RUM
H2P_Msg_Addr	x'7801 0050'	PCI Host to PowerPC Message Address Register (Reset Status)	R&RS
	x'7801 0060'	PCI Host to PowerPC Message Address Register	R/W
H2P_Doorbell	x'7801 0030'	PCI Host to PowerPC Doorbell Register (PCI Host Access)	R/SUM
	x'7801 0038'	PCI Host to PowerPC Doorbell Register (PowerPC Access)	R/RUM
E2P_Msg_Resource	x'7801 0040'	EPC to PowerPC Message Resource Register	R/W
E2P_Msg_Addr	x'7801 0048'	EPC to PowerPC Message Address Register	R
E2P_Doorbell	x'7801 0058'	EPC to PowerPC Doorbell Register (PowerPC Access)	R/RUM
P2E_Msg_Addr	x'7801 0068'	PowerPC to EPC Message Address Register	R/W
P2E_Doorbell	x'7801 0070'	PowerPC to EPC Doorbell Register (PowerPC Access)	R/SUM
E2H_Msg_Resource	x'7801 0080'	EPC to PCI Host Message Resource Register	R/W
E2H_Msg_Addr	x'7801 0088'	EPC to PCI Host Message Address Register	R
E2H_Doorbell	x'7801 0098'	EPC to PCI Host Doorbell Register (PCI Host Access)	R/RUM
H2E_Msg_Addr	x'7801 00A8'	PCI Host to EPC Message Address Register	R/W
Msg_Status	x'7801 00A0'	Message Status Register	R
H2E_Doorbell	x'7801 00B0'	PCI Host to EPC Doorbell Register (PCI Host Access)	R/SUM
SEAR	x'7801 00B8'	Slave Error Address Register	R

Symbolic Address	PLB Address	Description	Access
SESR	x'7801 00C0'	Slave Error Status Register	R
Parity_Error_Cntr	x'7801 00C8'	Parity Error Counter Register	R
PwrPC_Inst_Store	x‘0000 0000’ - x‘07FF FFFF’	PowerPC Instruction DRAM	R/W
Unassigned addresses in the range x‘7801 0000’ - x‘7801 FFFF’ are reserved			

10.6 CAB Address Map

Some components of the embedded PowerPC are also accessible via the NP4GS3's CAB Interface. These components are accessed using CAB addresses as shown in the CAB Address Map.

Symbolic Address	CAB Address	Description	Access
Mailbox and DRAM Interface Macro			
Boot_Redir_Inst	x'3800 0110' - x'3800 0117'	Boot Redirection Instruction Registers for instruction addresses x'FFFF FFE0' - x'FFFF FFFC'	R/W
E2P_Msg_Resource	x'3801 0010'	EPC to PowerPC Message Resource Register	R&RS
E2P_Msg_Addr	x'3801 0020'	EPC to PowerPC Message Address Register	R/W
E2P_Doorbell	x'3801 0040'	EPC to PowerPC Doorbell Register (PowerPC Access)	R/SUM
P2E_Msg_Addr	x'3801 0080'	PowerPC to EPC Message Address Register	R
	x'3802 0010'	PowerPC to EPC Message Address Register	R&RS
P2E_Doorbell	x'3801 0100'	PowerPC to EPC Doorbell Register (PowerPC Access)	R/RUM
E2H_Msg_Resource	x'3801 0200'	EPC to PCI Host Message Resource Register	R&RS
E2H_Msg_Addr	x'3801 0400'	EPC to PCI Host Message Address Register	R/W
E2H_Doorbell	x'3801 0800'	EPC to PCI Host Doorbell Register (PCI Host Access)	R/SUM
H2E_Msg_Addr	x'3801 1000'	PCI Host to EPC Message Address Register	R
	x'3802 0020'	PCI Host to EPC Message Address Register	R&RS
H2E_Doorbell	x'3801 2000'	PCI Host to EPC Doorbell Register (PCI Host Access)	R/RUM
Msg_Status	x'3801 4000'	Message Status Register	R

10.7 CAB Interface Macro

The CAB Interface macro provides duplicate facilities to support independent CAB addressing by the PCI Host processor and the embedded PowerPC. Exclusive access to these facilities, if required, is enforced through software discipline.

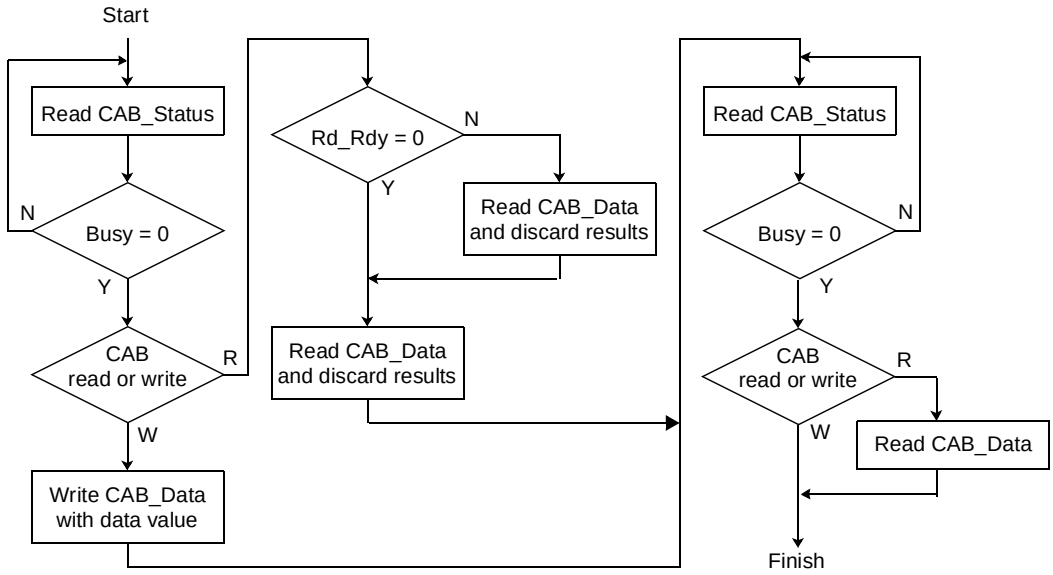
The PCI Host Processor can access the NP4GS3's CAB Interface through the following mechanism: after PCI configuration, one or more ranges of PCI addresses are mapped to PLB addresses. Accessing these PCI addresses also accesses PowerPC PLB resources which include the following CAB interface registers:

- CAB Address register is set to the value of the CAB address to be read or written.
- CAB Control register is written with parameters that control the behavior for CAB access.
- CAB Data register, when accessed, initiates a CAB access and determines its type (read or write).
- Status register is read to determine whether the CAB interface is waiting for a response (busy), whether the CAB is locked by the user and, for polled access, whether read data is ready (rd_rdy).

The CAB Control register (w/$\overline{p}$) controls the two modes of CAB access:

- Wait access, w/$\overline{p}$ = '1', causes the CAB Interface macro to insert wait states on the PLB until the CAB access is complete. Software need not read the CAB Status register to determine completion.
- Polled access, w/$\overline{p}$ = '0', requires software to read the CAB Status register to determine when a read or write access is complete (rd_rdy = '0' and busy = '0', respectively) before initiating another CAB transaction. A subsequent read of the CAB Data register initiates a CAB read access from the CAB address contained in the CAB Address register (busy = '1'). The data returned as a result of this read of the CAB_Data register is discarded. Software must then poll the CAB Status register until the CAB access is complete (busy = '0'). The CAB Status register indicates that the access is complete and data is waiting in the CAB Data register (busy = '0' and rd_rdy = '1'). A final read of the CAB Data returns the data

Figure 72: Polled Access Flow Diagram

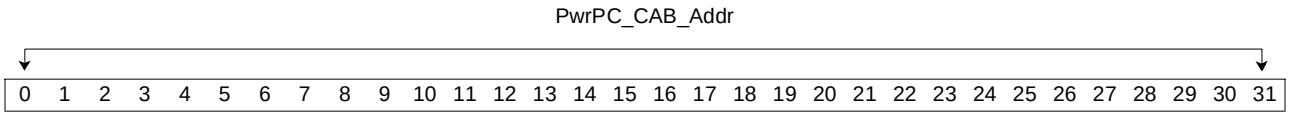


10.7.1 PowerPC CAB Address (PwrPC_CAB_Addr) Register

The PowerPC CAB Address register is accessible from the PLB and supplies a CAB address value for PowerPC access to NP4GS3 structures via the CAB Interface.

Access Type Read/Write

Base Address (PLB) x‘7800 0000’



Field Name	Bit(s)	Reset	Description
PwrPC_CAB_Addr	0:31	x’0000 0000’	CAB address value for PowerPC access to NP4GS3 structures via the CAB Interface.

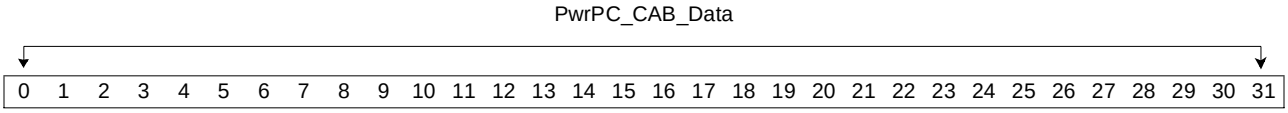
10.7.2 PowerPC CAB Data (PwrPC_CAB_Data) Register

The PowerPC CAB Data register is accessible from the PLB and contains the value of CAB data written or read when the PowerPC accesses NP4GS3 structures via the CAB Interface. Writing to the PwrPC_CAB_Data register has the side effect of initiating a write access on the CAB. The data value written to the PwrPC_CAB_Data register is written to the CAB address contained in the PwrPC_CAB_Addr register.

When the PwrPC_CAB_Cntl register has been configured with its w/$\overline{p}$ bit set to ‘1’ or if its w/$\overline{p}$ bit is set to ‘0’ and the rd_rdy bit of the PwrPC_CAB_Status register is set to ‘0’, a read of the PwrPC_CAB_Data register has the side effect of initiating a corresponding read access on the CAB. At the end of the CAB read access, the data value indicated by the PwrPC_CAB_Addr register is stored in the PwrPC_CAB_Data register and the rd_rdy bit of the PwrPC_CAB_Status register is set to '1. When the w/$\overline{p}$ bit set to ‘1’, the data value is also returned to the PLB. Otherwise, a subsequent read access of the PwrPC_CAB_Data register is required to retrieve the data value.

Access Type Read/Write

Base Address (PLB) x‘7800 0008’



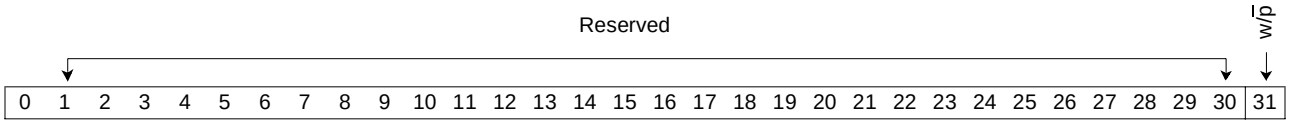
Field Name	Bit(s)	Reset	Description
PwrPC_CAB_Data	0:31	x’0000 0000’	CAB data written or read when the PowerPC accesses NP4GS3 structures via the CAB Interface.

10.7.3 PowerPC CAB Control (PwrPC_CAB_Cntl) Register

The PowerPC CAB Control register is accessible from the PLB and controls the CAB access protocol used by the PowerPC. The bit in this register indicates whether the wait or polled protocol is used when the PowerPC accesses CAB connected structures within the NP4GS3.

Access Type Read/Write

Base Address (PLB) x‘7800 0010’



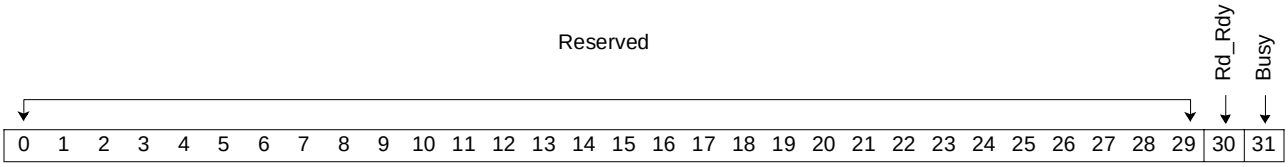
Field Name	Bit(s)	Reset	Description
Reserved	0:30		Reserved
w/$\overline{p}$	31	0	Wait or polled access control value 0 PowerPC polls the PwrPC_CAB_Status register to determine when access is complete 1 CAB Interface macro inserts wait states on the PLB until the CAB access is complete

10.7.4 PowerPC CAB Status (PwrPC_CAB_Status) Register

The PowerPC CAB Status register is accessible from the PLB and monitors the status of PowerPC CAB accesses. Bits within this register indicate the status of PowerPC accesses of CAB connected structures within the NP4GS3.

Access Type Read/Write

Base Address (PLB) x‘7800 0018’



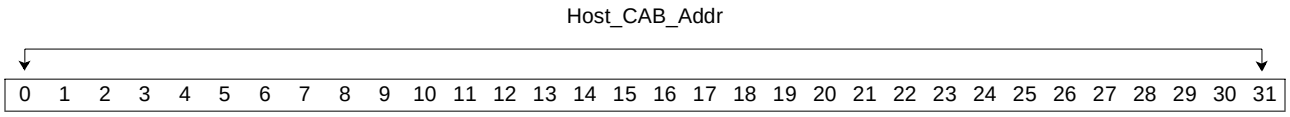
Field Name	Bit(s)	Reset	Description
Reserved	0:29		Reserved
Rd_Rdy	30	0	Read Data Ready indicator (used for polled access mode w/$\overline{p}$ = ‘0’ only). 0 No data in CAB Data register, if Busy = ‘0’, a new CAB access can begin 1 Data from a CAB read access is waiting in the CAB Data register
Busy	31	0	Busy indicator value. 0 CAB access is not pending. Set when the access is complete. 1 CAB access is pending. Set when an access command (read or write) is initiated.

10.7.5 PCI Host CAB Address (Host_CAB_Addr) Register

The PCI Host CAB Address register is accessible from the PLB and supplies a CAB address value for PCI Host access to NP4GS3 structures via the CAB Interface.

Access Type Read/Write

Base Address (PLB) x‘7800 8000’



Field Name	Bit(s)	Reset	Description
Host_CAB_Addr	0:31	x’0000 0000’	CAB address value for PCI Host access to NP4GS3 structures via the CAB Interface.

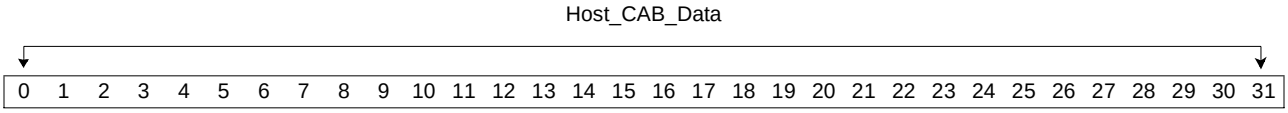
10.7.6 PCI Host CAB Data (Host_CAB_Data) Register

The PCI Host CAB Data register is accessible from the PLB and contains the value of CAB data written or read when the PCI Host accesses NP4GS3 structures via the CAB Interface. Writing to the Host_CAB_Data register has the side effect of initiating a write access on the CAB. The data value written to the Host_CAB_Data register is written to the CAB address contained in the Host_CAB_Addr register.

When the Host_CAB_Cntl register has been configured with its w/$\overline{p}$ bit set to ‘1’ or if its w/$\overline{p}$ bit is set to ‘0’ and the Rd_Rdy bit of the Host_CAB_Status register is set to ‘0’, a read of the Host_CAB_Data register has the side effect of initiating a corresponding read access on the CAB. At the end of the CAB read access, the data value indicated by the Host_CAB_Addr register is stored in the Host_CAB_Data register and the rd_rdy bit of the Host_CAB_Status register is set to '1. When the w/$\overline{p}$ bit set to ‘1’, the data value is also returned to the PLB. Otherwise, a subsequent read access of the Host_CAB_Data register is required to retrieve the data value.

Access Type Read/Write

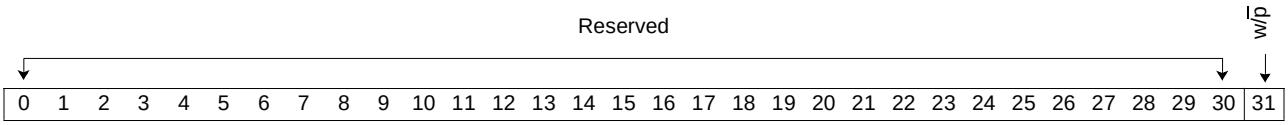
Base Address (PLB) x‘7800 8008’



Field Name	Bit(s)	Reset	Description
Host_CAB_Data	0:31	x’0000 0000’	CAB data written or read when the PCI Host accesses NP4GS3 structures via the CAB Interface.

10.7.7 PCI Host CAB Control (Host_CAB_Cntl) Register

The PCI Host Control register is accessible from the PLB and controls the CAB access protocol used by the PCI Host. The bit in this register indicates whether the wait or polled protocol is used when the PCI Host accesses CAB connected structures within the NP4GS3.



Access Type Read/Write

Base Address (PLB) x‘7800 8010’

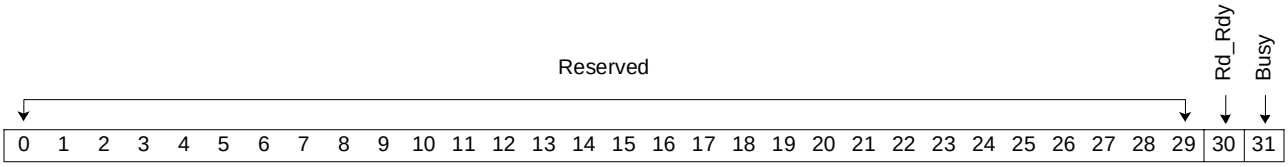
Field Name	Bit(s)	Reset	Description
Reserved	0:30		Reserved
w/$\overline{p}$	31	0	Wait or polled access control value 0 PCI Host polls the Host_CAB_Status register to determine when access is complete 1 CAB Interface macro inserts wait states on the PLB until the CAB access is complete

10.7.8 PCI Host CAB Status (Host_CAB_Status) Register

The PCI Host CAB Status register is accessible from the PLB and monitors the status of PCI Host CAB accesses. Bits within this register indicate the status of PCI Host accesses of CAB connected structures within the NP4GS3.

Access Type Read/Write

Base Address (PLB) x‘7800 8018’



Field Name	Bit(s)	Reset	Description
Reserved	0:29		Reserved
Rd_Rdy	30	0	Read Data Ready indicator (used for polled access mode w/$\overline{p}$ = ‘0’ only). 0 No data in CAB Data register, if Busy = ‘0’, a new CAB access can begin 1 Data from a CAB read access is waiting in the CAB Data register
Busy	31	0	Busy indicator value. 0 CAB access is not pending. Set when the access is complete. 1 CAB access is pending. Set when an access command (read or write) is initiated.

10.8 Mailbox Communications and DRAM Interface Macro

The Mailbox and DRAM Interface macro consists of two major portions. The first portion is a set of facilities for constructing and signalling messages between the various processors. A set of message resource registers allocates buffers for message construction, a set of message address registers accomplishes message signalling, and a set of doorbell registers accomplishes other inter-processor signalling. The second portion is hardware that interfaces the NP4GS3's DRAM controller to the PLB. The DRAM Interface maps a range of PLB addresses into DRAM addresses. DRAM connected via this interface stores PowerPC instructions, message data, and other data associated with the PowerPC.

The Mailbox and DRAM Interface macro also provides redirection of boot code. This function is used in system implementations in which the NP4GS3 does not boot from PCI memory. In these cases, hardware decodes the first PowerPC instruction fetch and supplies up to eight instructions from registers internal to the Mailbox and DRAM Interface. These registers are accessible via the CAB and are loaded by software prior to releasing the PowerPC's reset (PwrPC_Reset). Code stored in these registers redirects execution to locations in the DRAM.

10.8.1 Mailbox Communications Between PCI Host and PowerPC

Communication between the PCI Host and the PowerPC is accomplished through PCI Host to PowerPC (H2P) and PowerPC to PCI Host (P2H) interrupts. The P2H interrupt is implemented by asserting the NP4GS3's INTA# signal output. PCI interrupts are level sensitive and are asynchronous with the PCI_Clk signal. The existing PCI Macro's INTA# signal is supplemented with other interrupt generation outside of the PCI Macro. Using the PCI Macro, the PowerPC can send an interrupt to the PCI Host by writing to the PCI/PLB Macro's PCI Interrupt Control/Status register. This interrupt signal is recorded in the PCI Interrupt Status register. Doorbell and Message register operations are additional sources of PCI interrupts. The PCI Macro can interrupt the PowerPC by setting bit 13 of the PCI Macro's Bridge Options 2 register. This interrupt signal is applied to the PowerPC's Universal Interrupt Controller (UIC).

Communications between the PCI Host and the PowerPC use message buffers in PCI address space. Software running in the PCI Host manages these buffers. For communications from the PowerPC to the PCI Host, the starting address of empty message buffers are stored in the P2H Message Resource (P2H_Msg_Resource) register. This register is accompanied by a P2H_Bufr_Valid status flag, located in the Msg_Status register, that indicates whether or not the P2H_Msg_Resource register contains a valid buffer address.

The PCI Host writes the P2H_Msg_Resource register with the PCI address of an empty message buffer and the valid indicator flag is set. The PowerPC reads the flag value when a message buffer is required and then reads the valid message buffer address value from the P2H_Msg_Resource register. Reading the P2H_Msg_Resource register resets the valid indicator bit. By polling this indicator bit, the PCI Host knows when to replenish the P2H_Msg_Resource register with a new buffer address value.

Having acquired a message buffer, the PowerPC composes a message in the buffer and writes the buffer's starting address value into the P2H_Msg_Addr register. This write also sets the PCI_Interr_Status register's P2H_msg bit. A PCI interrupt is generated when the corresponding bit of the PCI_Interr_Ena register is set. The PCI Host reads the P2H_Msg_Addr register to find and process the message. The read clears the interrupt condition.

For communication from the PCI Host to the PowerPC, messages are composed in buffers in the PCI address space. Each message is then signalled to the PowerPC by writing its starting PCI address value into the H2P_Msg_Addr register. Writing this register sets the H2P_Msg_Interr to the PowerPC's UIC and sets the H2P_Msg_Busy bit of the Msg_Status register. An interrupt is generated when enabled by the UIC. The PowerPC reads the H2P_Msg_Addr register at one address location to find and process the message and,

due to the read, the interrupt condition is cleared. A subsequent read of the H2P_Msg Addr register at another address location will reset the H2P_Msg_Busy bit of the Msg_Status register. This second read signals the PCI Host that the PowerPC has finished processing the data buffer, allowing it to be reused.

The P2H_Msg_Addr register is written by the PowerPC and read by the PCI Host processor. Whenever the PowerPC writes the P2H_Msg_Addr register, a bit in the PCI Interrupt Status register is set to '1' and is independent of the value written. The PCI Interrupt Status register indicates the source of the PCI Interrupt. The PCI Interrupt Status register bit is reset to '0' when the PCI Host processor reads the P2H_Msg_Addr register. Software discipline controls the setting of the interrupt by the PowerPC and the resetting of the interrupt by the PCI Host (only the PowerPC writes this register and only the PCI Host reads this register).

The Doorbell register is written and read by either the PowerPC or the PCIHost processor. The value recorded in this register depends upon the data value to be written, the current contents of the register, and whether the PowerPC or the PCI Host processor is writing the register.

When written by the PowerPC, each bit of the register's current contents is compared with the corresponding data bit to be written. If the value of the data bit is '1', then the corresponding Doorbell register is set to '1'. Otherwise it remains unchanged ('0' or '1'). This effect is referred to as set-under-mask (SUM).

When written by the PCI Host processor, each bit of the register's current contents is compared with the corresponding data bit to be written. If the value of the data bit is '1', then the corresponding Doorbell register bit is reset to '0'. Otherwise, it remains unchanged ('0' or '1'). This effect is referred to as reset-under-mask (RUM). If one or more of the bits in the Doorbell register are '1', then a signal is generated and stored in the PCI Interrupt Status register.

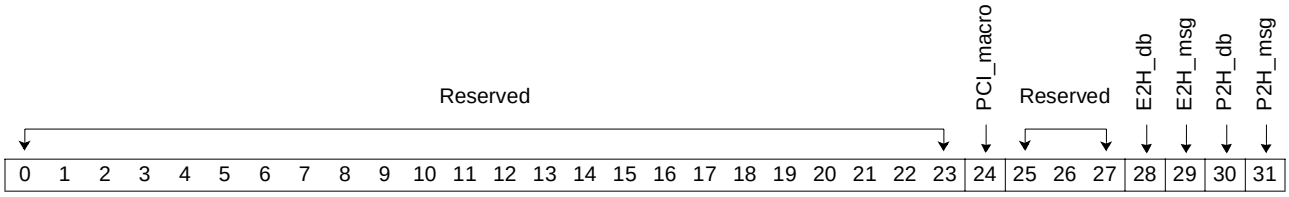
Any of the signals recorded as '1' in the PCI Interrupt Status register activates the INTA# signal if the corresponding condition is enabled in the NP4GS3's PCI Interrupt Enable register. The PCI Host processor reads the PCI Interrupt Status register to determine the interrupt source.

10.8.2 PCI Interrupt Status (PCI_Interr_Status) Register

The PCI Interrupt Status register is accessible from the PLB and records the source of the PCI interrupts generated by the NP4GS3. This register's bits are set by hardware and are read by PCI Host software. When the interrupt source is cleared, the corresponding bit of the PCI_Interr_Status register is also cleared.

Access Type Read Only

Base Address (PLB) x'7801 0000'



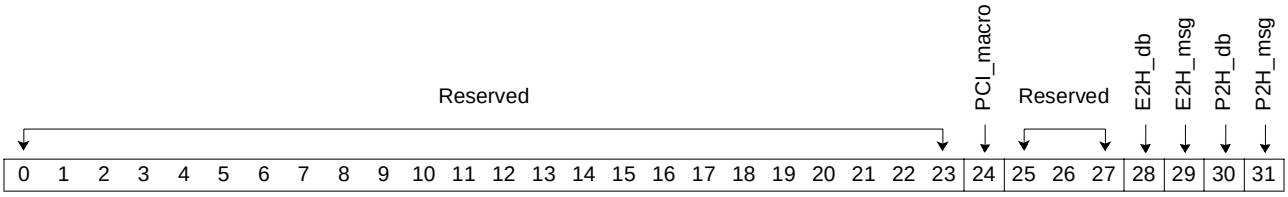
Field Name	Bit(s)	Reset	Description
Reserved	0:23		Reserved
PCI_macro	24	0	PCI interrupt from macro indicator. 0 Interrupt absent 1 Interrupt present
Reserved	25:27		Reserved
E2H_db	28	0	EPC to PCI Host doorbell indicator. 0 PCI interrupt from E2H_Doorbell register absent 1 PCI interrupt from E2H_Doorbell register present
E2H_msg	29	0	EPC to PCI Host message indicator. 0 PCI interrupt from E2H_Message register absent 1 PCI interrupt from E2H_Message register present
P2H_db	30	0	PowerPC to PCI Host doorbell indicator. 0 PCI interrupt from P2H_Doorbell register absent 1 PCI interrupt from P2H_Doorbell register present
P2H_msg	31	0	PowerPC to PCI Host message indicator. 0 PCI interrupt from P2H_Message register absent 1 PCI interrupt from P2H_Message register present

10.8.3 PCI Interrupt Enable (PCI_Interr_Ena) Register

The PCI Interrupt Enable register is accessible from the PLB and enables PCI interrupts from sources within the NP4GS3.

Access Type Read/Write

Base Address (PLB) x‘7801 0008’



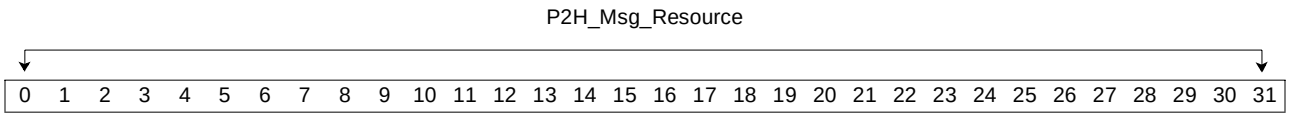
Field Name	Bit(s)	Reset	Description
Reserved	0:23		Reserved
PCI_macro	24	0	PCI macro interrupt - controls the assertion of a PCI interrupt from the PCI macro. 0 Interrupt disabled 1 Interrupt enabled
Reserved	25:27		Reserved
E2H_db	28	0	EPC to PCI Host doorbell interrupt - controls the assertion of a PCI interrupt from the E2H_Doorbell register. 0 Interrupt disabled 1 Interrupt enabled
E2H_msg	29	0	EPC to PCI Host message interrupt - controls the assertion of a PCI interrupt from the E2H_Message register. 0 Interrupt disabled 1 Interrupt enabled
P2H_db	30	0	PowerPC to PCI Host doorbell interrupt - controls the assertion of a PCI interrupt from the P2H_Doorbell register. 0 Interrupt disabled 1 Interrupt enabled
P2H_msg	31	0	PowerPC to PCI Host message interrupt - controls the assertion of a PCI interrupt from the P2H_Message register. 0 Interrupt disabled 1 Interrupt enabled

10.8.4 PowerPC to PCI Host Message Resource (P2H_Msg_Resource) Register

The PowerPC to PCI Host Message Resource register is accessible from the PLB. The PowerPC uses this register to obtain message buffers in PCI address space for messages the PowerPC sends to the PCI Host processor. The PCI Host writes the starting PCI address value of a message buffer into the P2H_Msg_Resource register. Writing to this register sets the P2H_Bufr_Valid flag and reading the P2H_Msg_Resource register clears this flag.

Access Type Read/Write

Base Address (PLB) x‘7801 0010’



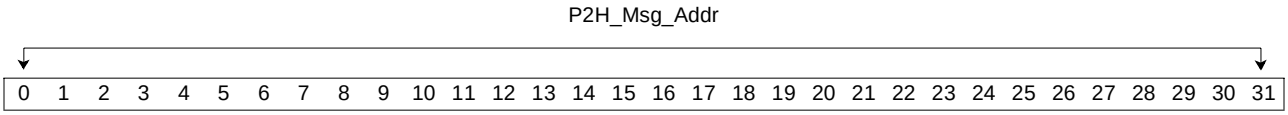
Field Name	Bit(s)	Reset	Description
P2H_Msg_Resource	0:31	x’0000 0000’	PowerPC to PCI Host Message Resource value, written with the PCI starting address of a message buffer.

10.8.5 PowerPC to Host Message Address (P2H_Msg_Addr) Register

The PowerPC to PCI Host Message Address register is accessible from the PLB and is used by the PowerPC to send messages to the PCI Host processor. The value written into this register is the PCI address at which the message begins. Writing to this register sets the P2H_msg bit of the PCI_Interr_Status register. When the corresponding bit of the PCI_Interr_Ena register is set to '1', the INTA# signal of the PCI bus is activated. The P2H_msg bit of the PCI_Interr_Status register is reset when the interrupt service routine reads the P2H_Msg_Addr register.

Access Type Read/Write

Base Address (PLB) x'7801 0018'

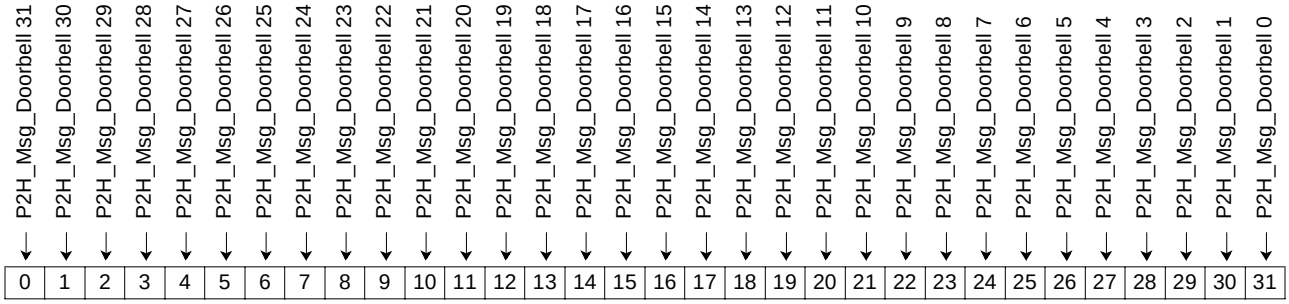


Field Name	Bit(s)	Reset	Description
P2H_Msg_Addr	0:31	x'0000 0000'	PowerPC to PCI Host Message address value, indicates the PCI starting address of a message.

10.8.6 PowerPC to Host Doorbell (P2H_Doorbell) Register

The PowerPC to PCI Host Doorbell register is accessible from the PLB and is used by the PowerPC to signal interrupts to the PCI Host processor. The PowerPC has read and SUM write access to this register. The data contains the mask used to access this register. When bits of this register are set to '1' and the corresponding bit of the PCI_Interr_Ena register is set to '1', the INTA# signal of the PCI bus is activated. The PCI Host processor reads this register to determine which of the doorbells have been activated. The PCI Host processor has read and RUM write access to this register using a different PLB address value.

Access Type	Power PC	x'7801 0020'
	Host	x'7801 0028'
Base Address (PLB)	Power PC	Read/Set-Under-Mask
	Host	Read/Reset-Under-Mask

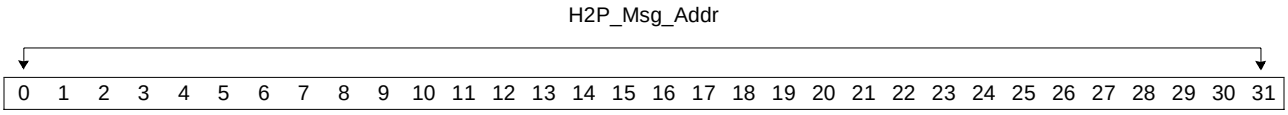


Field Name	Bit(s)	Reset	Description
P2H_Msg_Doorbell 31	0	0	PowerPC to PCI Host Doorbell - indicates which of the 32 possible doorbells have been activated. 0 Not activated 1 Activated
P2H_Msg_Doorbell 30:1	1:30	0 for all	
P2H_Msg_Doorbell 0	31	0	

10.8.7 Host to PowerPC Message Address (H2P_Msg_Addr) Register

The PCI Host to PowerPC Message Address register is accessible from the PLB and is used by the PCI Host to send messages to the PowerPC processor. The value written into this register is a message's PCI starting address. Writing to this register activates the H2P_Msg_Interr input to the PowerPC's UIC. When this interrupt is enabled, an interrupt to the PowerPC is generated. Reading this register resets the H2P_Msg_Interr input to the UIC. Reading this register at an alternate PLB address resets the Msg_Status register's H2P_Msg_Busy bit.

Access Type	1	Read and Reset Status
	2	Read/Write
Base Address (PLB)	1	x'7801 0050'
	2	x'7801 0060'



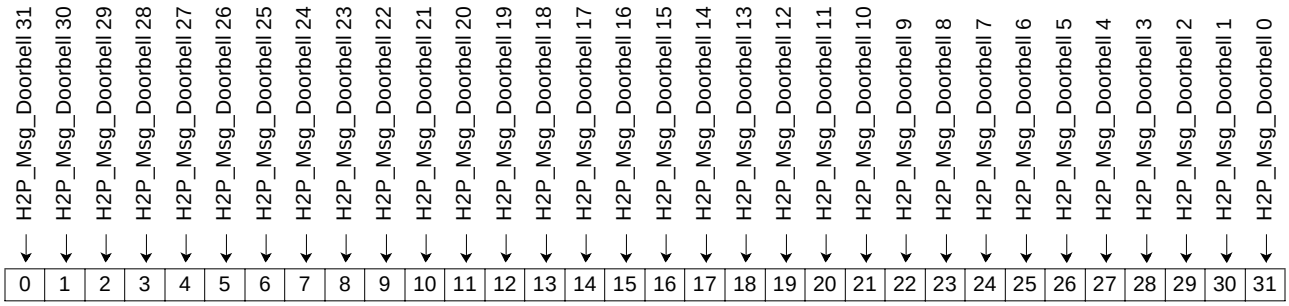
Field Name	Bit(s)	Reset	Description
H2P_Msg_Addr	0:31	x'0000 0000'	The value is a message's PCI starting address.

10.8.8 Host to PowerPC Doorbell (H2P_Doorbell) Register

The PCI Host to PowerPC Doorbell (H2P_Doorbell) register is accessible from the PLB and is used by the PCI Host processor to signal interrupts to the PowerPC. The PCI Host processor has read and SUM write access to this register. The data contains the mask used to access this register. When any of this register's bits are set to '1', an interrupt signal of the PowerPC's UIC is activated. The PowerPC reads this register to determine which of the doorbells have been activated. The PowerPC has read and RUM write access to this register using a different PLB address value.

Access Type	Host	Read/Set-Under-Mask Write
	PowerPC	Read/Reset-Under-Mask Write
Base Address (PLB)	Host	x'7801 0030'
	PowerPC	x'7801 0038'

H2P_Doorbells



Field Name	PLB Bit(s)	Reset	Description
H2P_Msg_Doorbell 31	0	0	PCI Host to PowerPC Doorbell - indicates which of the 32 possible doorbells have been activated. 0 Not activated 1 Activated
H2P_Msg_Doorbell 30:1	1:30	0 for all	
H2P_Msg_Doorbell 0	31	0	

10.8.9 Mailbox Communications Between PowerPC and EPC

Communication between the PowerPC and the EPC is accomplished by writing message data into buffers in the PowerPC's DRAM and signalling the destination processor with an interrupt. The PowerPC software manages message data buffers for PowerPC to EPC (P2E) messages and EPC to PowerPC (E2P) messages.

Message data buffers are allocated to the EPC by writing the buffers' starting address into the E2P_Msg_Resource register. Writing to this register sets the E2P_Bufr_Valid flag in the Msg_Status register. This flag indicates to the EPC that the buffer is valid and can be used by the EPC. The EPC reads the E2P_Bufr_Valid value when a message buffer is required. The EPC then reads the E2P_Msg_Resource register via the CAB Interface to obtain the address of the message buffer and the E2P_Bufr_Valid indicator bit is reset. By polling this indicator bit, the PowerPC knows when to replenish the E2P_Msg_Resource register with a new buffer address value. Having acquired a message data buffer, the EPC composes a message in the buffer and writes the buffer's starting address value into the E2P_Msg_Addr register. Writing to this register generates an interrupt signal to the PowerPC's UIC. The PowerPC reads this register to find and process the message and, due to the read, the interrupt condition is cleared.

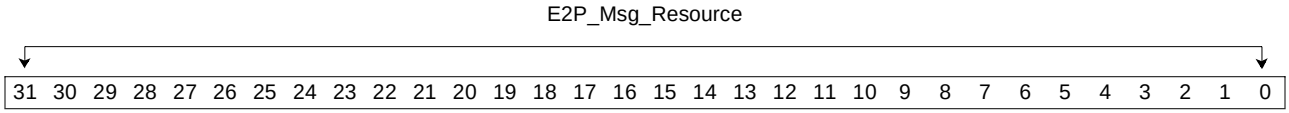
There is no need for a P2E_Msg_Resource register because the PowerPC software manages the message data buffers The PowerPC composes a message in one of the data buffers and writes the starting address of the buffer into the P2E_Msg_Addr register. Writing to this register produces an interrupt to the EPC and sets the P2E_Msg_Busy bit of the Msg_Status register. As long as this flag is set, the EPC is processing the message buffer. The EPC reads the P2E_Msg_Addr register to locate the buffer in the PowerPC's DRAM. The EPC resets the P2E_Msg_Busy bit by reading the P2E_Msg_Address register at an alternate CAB address when message processing is complete. The PowerPC will poll the busy flag to determine when the buffer can be reused.

10.8.10 EPC to PowerPC Resource (E2P_Msg_Resource) Register

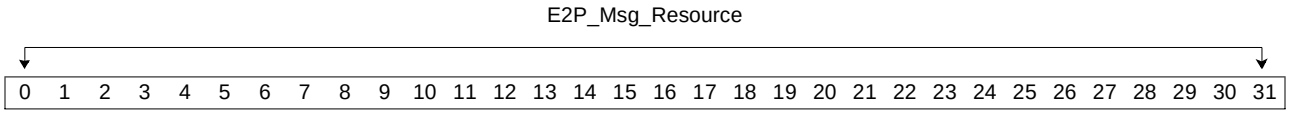
The PowerPC accesses the EPC to PowerPC Message Resource register from the PLB while the EPC accesses this register from its CAB Interface. The EPC uses this register to obtain message buffers in the PowerPC's DRAM address space for messages it sends to the PowerPC processor. The PowerPC writes the starting DRAM address value of a message buffer. Writing to this register sets the E2P_Bufr_Valid flag in the Msg_Status register. Reading the E2P_Msg_Resource register from the CAB resets this flag.

Access Type	CAB	Read
	PLB	Read/Write
Base Address	CAB	x'3801 0010'
	PLB	x'7801 0040'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Resource	31:0	x'0000 0000'	EPC to PowerPC Message Resource - written with the PowerPC's DRAM starting address of a message buffer.

PLB View

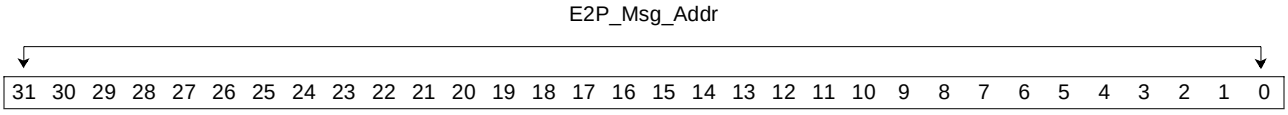
Field Name	Bit(s)	Reset	Description
E2P_Msg_Resource	0:31	x'0000 0000'	EPC to PowerPC Message Resource - written with the PowerPC's DRAM starting address of a message buffer.

10.8.11 EPC to PowerPC Message Address (E2P_Msg_Addr) Register

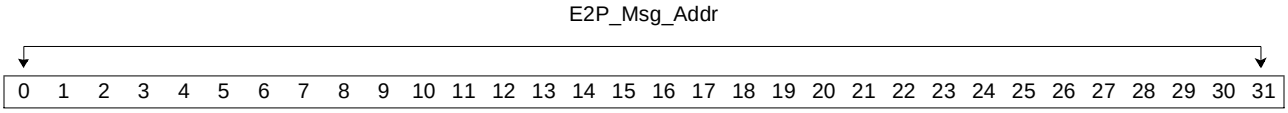
The PowerPC accesses the EPC to PowerPC Message Address register from the PLB while the EPC accesses this register from its CAB Interface. The EPC uses this register to send messages to the PowerPC processor. The value written into this register is the PowerPC's DRAM address at which the message begins. Writing to this register sets E2P_Msg_Interr input to the PowerPC's UIC. When the UIC is configured to enable this input, an interrupt signal to the PowerPC is activated. Reading the E2P_Msg_Addr register via the PLB address resets the E2P_Msg_Interr input to the PowerPC's UIC.

Access Type	CAB	Read/Write
	PLB	Read
Base Address	CAB	x'3801 0020'
	PLB	x'7801 0048'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Addr	0:31	x'0000 0000'	EPC to PowerPC Message Address - indicates the PCI starting address of a message.

PLB View

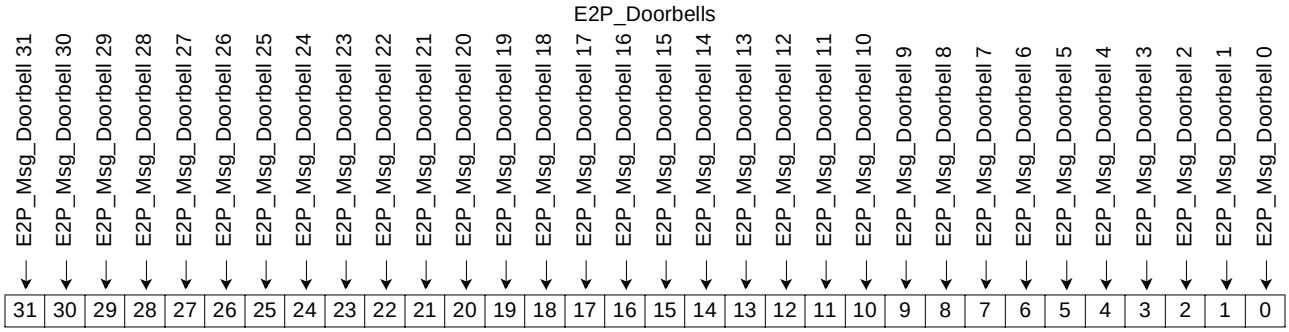
Field Name	Bit(s)	Reset	Description
E2P_Msg_Addr	0:31	x'0000 0000'	EPC to PowerPC Message Address - indicates the PCI starting address of a message.

10.8.12 EPC to PowerPC Doorbell (E2P_Doorbell) Register

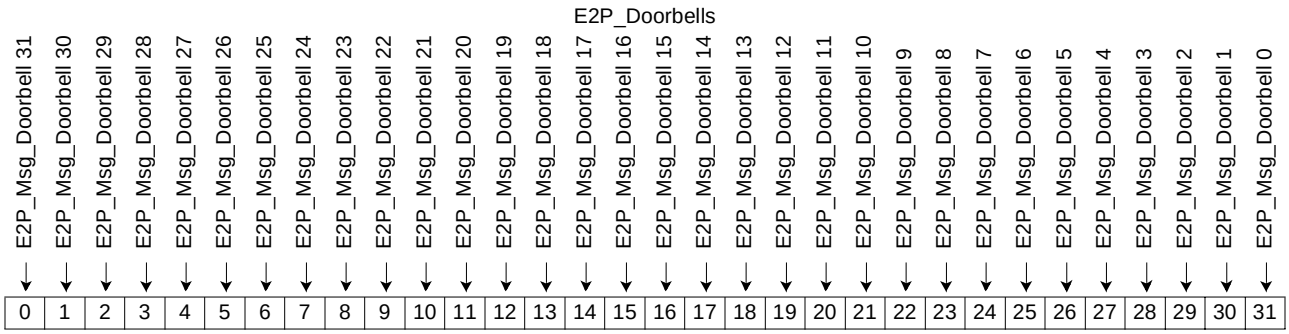
The PowerPC accesses the EPC to PowerPC Doorbell register from the PLB while the EPC accesses this register from the CAB Interface. The EPC uses this register to signal interrupts to the PowerPC. The EPC has read and SUM write access to this register using a CAB address value. The data contains the mask used to access this register. When any of this register's bits are set to '1' an interrupt signal to the PowerPC's UIC is activated. The PowerPC reads this register to determine which of the doorbells have been activated.The PowerPC has read and RUM write access to this register using a PLB address value.

Access Type	CAB	Read/Set-Under-Mask Write
	PLB	Read/Reset-Under-Mask Write
Base Address	CAB	x'3801 0040'
	PLB	x'7801 0058'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Doorbell 31	31	0	EPC to PowerPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
E2P_Msg_Doorbell 30:1	30:1	0 for all	
E2P_Msg_Doorbell 0	0	0	

PLB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Doorbell 31	0	0	EPC to PowerPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
E2P_Msg_Doorbell 30:1	1:30	0 for all	
E2P_Msg_Doorbell 0	31	0	

10.8.13 EPC Interrupt Vector Register

The EPC contains an Interrupt Vector 2 register which is accessible from the CAB Interface. This register records the source of EPC interrupts generated by the NP4GS3. This register's bits are set by hardware and are read by EPC software to determine the source of interrupts. (For more information on Interrupt Vector Registers, see Section 3 of the IBM Network Processor Hardware Reference Manual.)

10.8.14 EPC Interrupt Mask Register

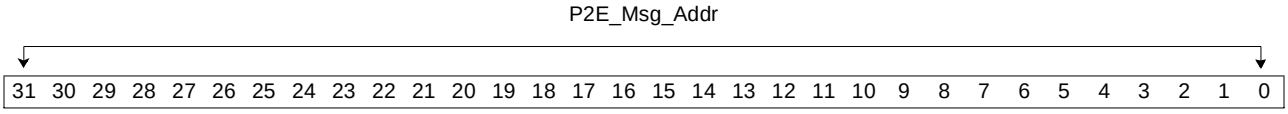
The EPC contains a Interrupt Mask 2 register which is accessible from the CAB Interface. This register enables EPC interrupts from sources within the NP4GS3. (For more information on Interrupt Mask Registers, see Section 3 of the IBM Network Processor Hardware Reference Manual.)

10.8.15 PowerPC to EPC Message Address (P2E_Msg_Addr) Register

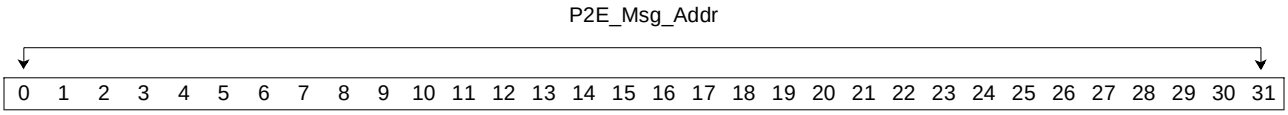
The PowerPC accesses the PowerPC to EPC Message Address register from the PLB while the EPC accesses this register from the CAB Interface. This register is used by the PowerPC to send messages to the EPC. The value written into this register is the PowerPC's DRAM address at which the message begins. Writing to this register sets the P2E_Msg_Interr signal to the EPC and the P2E_Msg_Busy bit of the Msg_Status register. Reading the P2E_Msg_Addr register from the CAB will reset the P2E_Msg_Interr signal to the EPC. Reading the P2E_Msg_Addr from an alternate CAB address will reset the P2E_Msg_Busy bit of the Msg_Status register.

Access Type	CAB 1	Read
	CAB 2	Read and Reset Status
	PLB	Read/Write
Base Address	CAB 1	x'3801 0080'
	CAB 2	x'3802 0010'
	PLB	x'7801 0068'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
P2E_Msg_Addr	31:0	x'0000 0000'	PowerPC to EPC Message Address - indicates a message's PowerPC DRAM starting address.

PLB View

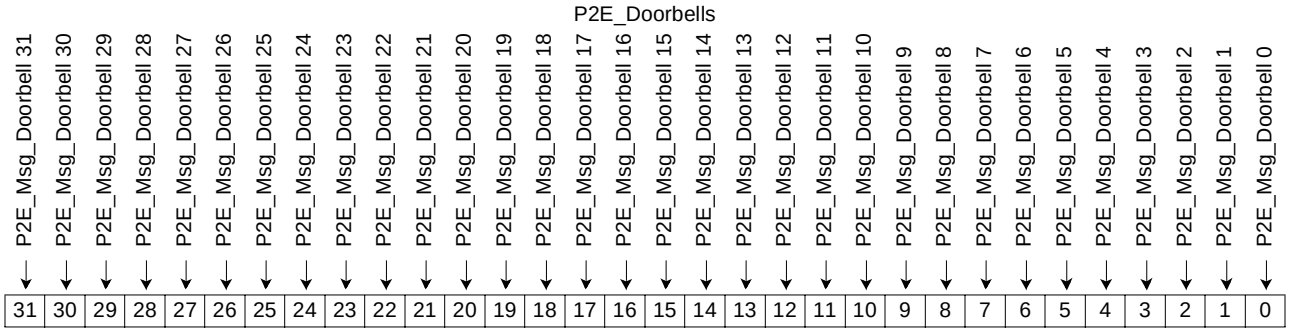
Field Name	Bit(s)	Reset	Description
P2E_Msg_Addr	0:31	x'0000 0000'	PowerPC to EPC Message Address - indicates a message's PowerPC DRAM starting address.

10.8.16 PowerPC to EPC Doorbell (P2E_Doorbell) Register

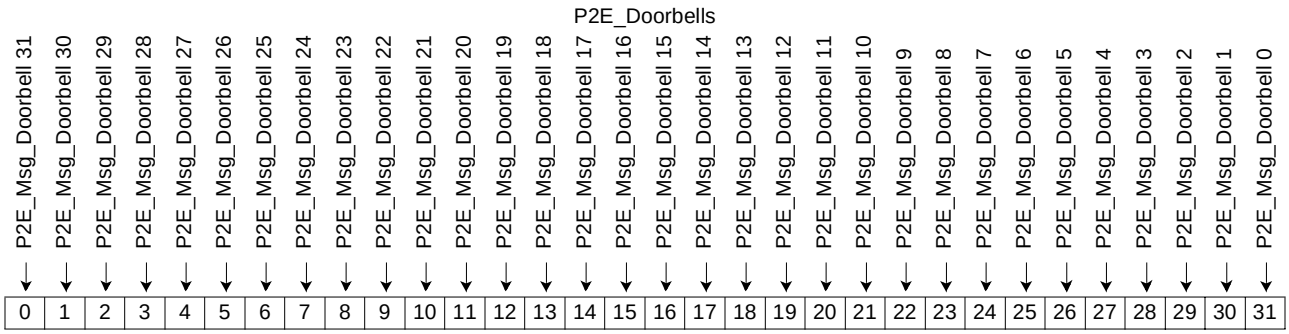
The PowerPC accesses the PowerPC to EPC Doorbell register from the PLB while the EPC accesses this register from the CAB Interface. The PowerPC uses this register to signal interrupts to the EPC. The PowerPC has read and SUM write access to this register. The data contains the mask used to access this register. When any of this register's bits are set to '1', an interrupt signal of the EPC is activated. This register is read by the EPC to determine which of the doorbells have been activated.The EPC has read and RUM write access to this register using a CAB address value.

Access Type	CAB	Read/Reset-Under-Mask Write
	PLB	Read/Set-Under-Mask Write
Base Address	CAB	x'3801 0100'
	PLB	x'7801 0070'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
P2E_Msg_Doorbell 31	31	0	PowerPC to EPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
P2E_Msg_Doorbell 30:1	30:1	0 for all	
P2E_Msg_Doorbell 0	0	0	

PLB View

Field Name	Bit(s)	Reset	Description
P2E_Msg_Doorbell 31	0	0	PowerPC to EPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
P2E_Msg_Doorbell 30:1	1:30	0 for all	
P2E_Msg_Doorbell 0	31	0	

10.8.17 Mailbox Communications Between PCI Host and EPC

Communication between the PCI Host processor and the EPC is accomplished by writing message data into buffers in the PowerPC's DRAM and signalling the destination processor with an interrupt. The PCI Host processor software manages message data buffers for PCI Host processor to EPC (H2E) messages and EPC to PCI Host processor (E2H) messages.

Message data buffers are allocated to the EPC by writing the buffers' starting address into the E2H_Msg_Resource register. Writing this register sets the E2H_Bufr_Valid flag in the Msg_Status register. This flag indicates to the EPC that the buffer is valid and can be used by the EPC. The EPC reads the E2H_Bufr_Valid value when a message buffer is required. The EPC then reads the E2H_Msg_Resource register via the CAB Interface and the E2P_Bufr_Valid indicator bit is reset. By polling this indicator bit, the PCI Host processor knows when to replenish the E2H_Msg_Resource register with a new buffer address value. Having acquired a message data buffer, the EPC will compose a message in the buffer and write the buffer's starting address value into the E2H_Msg_Addr register. Writing to this register generates an interrupt to the PCI Host processor. The PCI Host processor reads this register to find and process the message. Reading this register clears the interrupt condition.

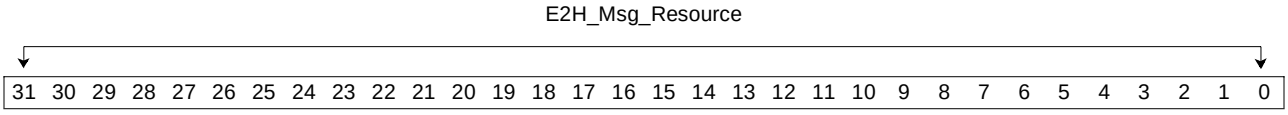
Since the PCI Host processor software manages the message data buffers, there is no need for an H2E_Msg_Resource register. The PCI Host processor composes a message in one of the data buffers and writes the starting address of the buffer into the H2E_Msg_Addr register. Writing to this register produces an interrupt input signal to the EPC and sets the H2E_Msg_Busy bit of the Msg_Status register. As long as this flag is set, the EPC is processing the message buffer. The EPC reads the H2E_Msg_Addr register to locate the buffer in the PowerPC's DRAM and reset the interrupt input signal to the EPC. The EPC resets the H2E_Msg_Busy bit by reading the P2E_Msg_Addr register from an alternate CAB address when message processing is complete. The PowerPC will poll the H2E_Msg_Busy bit to determine when the buffer can be reused.

10.8.18 EPC to PCI Host Resource (E2H_Msg_Resource) Register

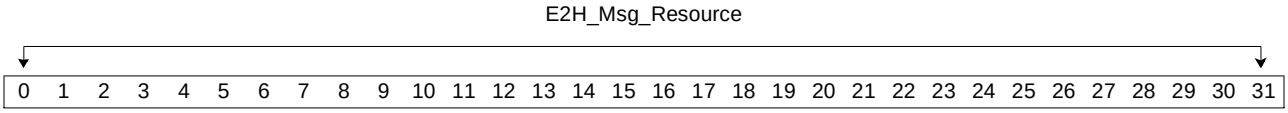
The PCI Host processor accesses the EPC to PCI Host Message Resource register from the PLB while the EPC accesses this register from its CAB Interface. This EPC uses this register to obtain message buffers in the PowerPC's DRAM address space for messages it sends to the PCI Host processor. The PCI Host processor writes the starting DRAM address value of a message buffer into the E2P_Msg_Resource register. Writing to this register sets the E2H_Bufr_Valid flag. Reading the E2H_Msg_Resource register from the CAB resets this flag.

Access Type	CAB	Read
	PLB	Read/Write
Base Address	CAB	x'3801 0200'
	PLB	x'7801 0080'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2H_Msg_Resource	31:0	x'0000 0000'	EPC to PCI Host Message Resource - written with the PowerPC's DRAM starting address of a message buffer.

PLB View

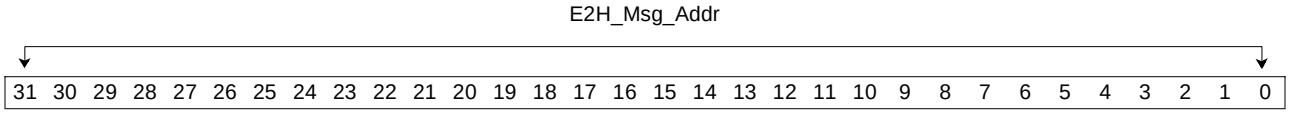
Field Name	Bit(s)	Reset	Description
E2H_Msg_Resource	0:31	x'0000 0000'	EPC to PCI Host Message Resource - written with the PowerPC's DRAM starting address of a message buffer.

10.8.19 EPC to PCI Host Message Address (E2H_Msg_Addr) Register

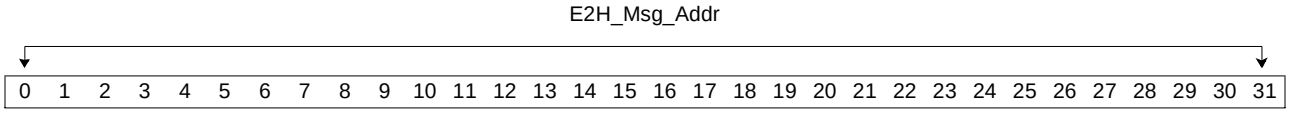
The PCI Host Processor accesses the EPC to PCI Host Message Address register from the PLB while the EPC accesses this register from its CAB Interface. The EPC uses this register to send messages to the PCI Host processor. The value written into this register is the PowerPC's DRAM address at which the message begins. Writing to this register sets the E2H_msg bit of the PCI_Interr_Status register. When the corresponding bit of the PCI_Interr_Ena register is set to '1', an interrupt signal to the PCI Host processor is activated. Reading the E2H_Msg_Addr register from the PLB resets the E2H_msg bit of the PCI_Interr_Status register.

Access Type	CAB	Read/Write
	PLB	Read
Base Address	CAB	x'3801 0400'
	PLB	x'7801 0088'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2H_Msg_Addr	31:0	x'0000 0000'	EPC to PCI Host Message Address - indicates the PowerPC's DRAM starting address of a message.

PLB View

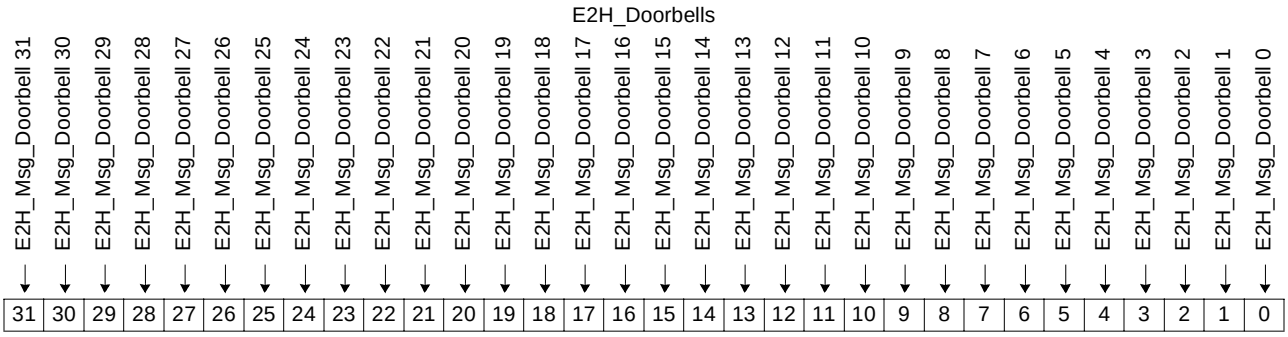
Field Name	Bit(s)	Reset	Description
E2H_Msg_Addr	0:31	x'0000 0000'	EPC to PCI Host Message Address - indicates the PowerPC's DRAM starting address of a message.

10.8.20 EPC to PCI Host Doorbell (E2H_Doorbell) Register

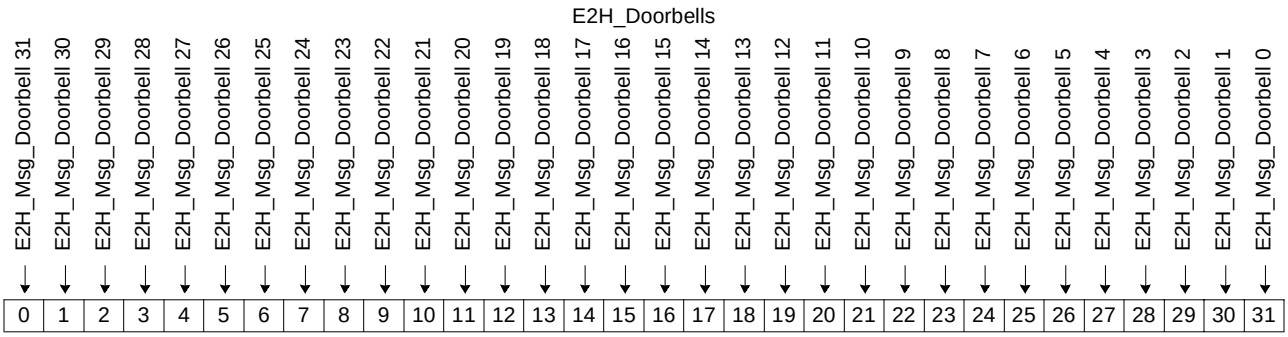
The PCI Host Processor accesses the EPC to PCI Host Doorbell register from the PLB while the EPC accesses this register from the CAB Interface. The EPC uses this register to signal interrupts to the PCI Host processor. The EPC has read and SUM write access to this register using a CAB address value. The mask used to access this register is contained in the data. When any of this register's bits are set to '1' and the corresponding bit of the PCI_Interr_Ena register is set to '1', an interrupt signal of the PCI Host processor is activated. This register is read by the PCI Host processor to determine which of the doorbells have been activated. The PCI Host processor has read and RUM write and read access to this register using a PLB address value.

Access Type	CAB	Read/Set_Under_Mask Write
	PLB	Read/Reset_Under_Mask Write
Base Address	CAB	x'3801 0800'
	PLB	x'7801 0098'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
E2H_Msg_Doorbell 31	31	0	EPC to PCI Host Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
E2H_Msg_Doorbell 30:1	30:1	0 for all	
E2H_Msg_Doorbell 0	0	0	

PLB View

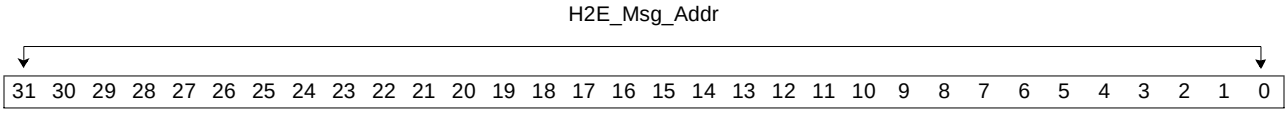
Field Name	Bit(s)	Reset	Description
E2H_Msg_Doorbell 31	0	0	EPC to PCI Host Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
E2H_Msg_Doorbell 30:1	1:30	0 for all	
E2H_Msg_Doorbell 0	31	0	

10.8.21 PCI Host to EPC Message Address (H2E_Msg_Addr) Register

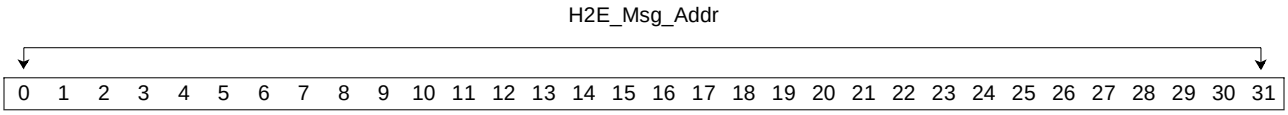
The PCI Host processor accesses the PCI Host to EPC Message Address register from the PLB while the EPC accesses this register from the CAB Interface. The PCI Host uses this register to send messages to the EPC. The value written into this register is the PowerPC's DRAM address at which the message begins. Writing to this register sets the H2E_Msg_Interr signal to the EPC and the H2E_Msg_Busy bit of the Msg_Status register. Reading the H2E_Msg_Addr register from the CAB will reset the H2E_Msg_Interr signal to the EPC. Reading the H2E_Msg_Addr from an alternate CAB will reset the H2E_Msg_Busy bit of the Msg_Status register.

Access Type	CAB 1	Read
	CAB 2	Read & Reset Status
	PLB	Read/Write
Base Address	CAB 1	x'3801 1000'
	CAB 2	x'3802 0020'
	PLB	x'7801 00A8'

CAB View



PLB View



CAB View

Field Name	Bit(s)	Reset	Description
H2E_Msg_Addr	31:0	x'0000 0000'	PCI Host to EPC Message Address - indicates a message's PowerPC DRAM starting address.

PLB View

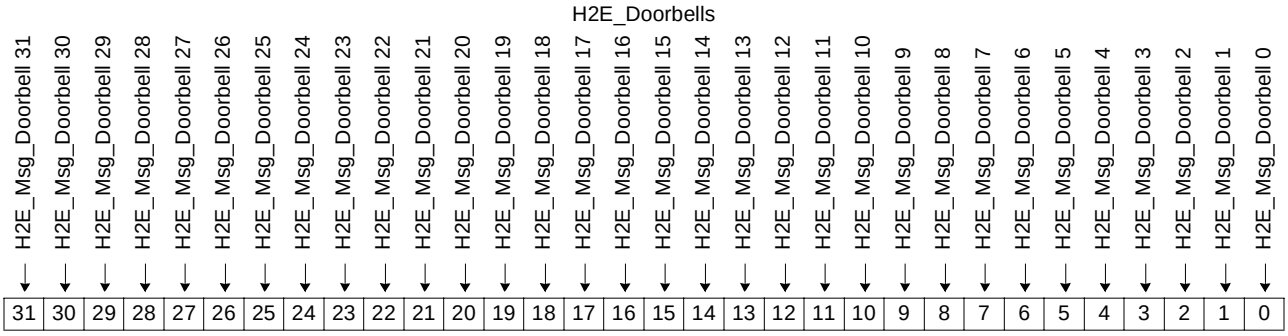
Field Name	Bit(s)	Reset	Description
H2E_Msg_Addr	0:31	x'0000 0000'	PCI Host to EPC Message Address - indicates a message's PowerPC DRAM starting address.

10.8.22 PCI Host to EPC Doorbell (H2E_Doorbell) Register

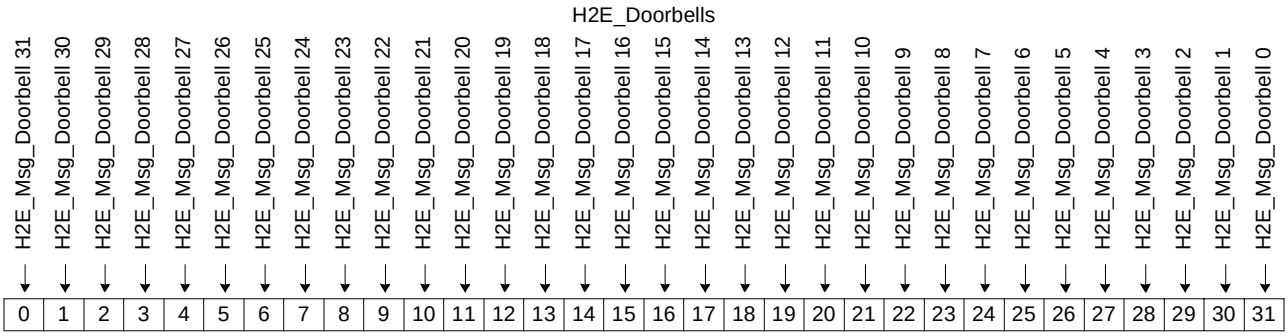
The PCI Host processor accesses the PCI Host to EPC Doorbell register from the PLB while the EPC accesses this register from the CAB Interface. The PCI Host processor uses this register to signal interrupts to the EPC. The PCI Host processor has read and SUM write access to this register. The data contains the mask used to access this register. When any of this register's bits are set to '1', an interrupt signal of the EPC is activated. This register is read by the EPC to determine which of the doorbells have been activated.The EPC has read and RUM write access to this register using a CAB address value.

Access Type	CAB	Read/Reset_Under_Mask Write
	PLB	Read/Set_Under_Mask Write
Base Address	CAB	x'3801 2000'
	PLB	x'7801 00B0'

CAB View



PLB View



CAB View

Field Name	Bit	Reset	Description
H2E_Msg_Doorbell 31	31	0	PCI Host to EPC Doorbell - indicates which of the 32 possible doorbells have been activated. 0 Not activated 1 Activated
H2E_Msg_Doorbell 30:1	30:1	0 for all	
H2E_Msg_Doorbell 0	0	0	

PLB View

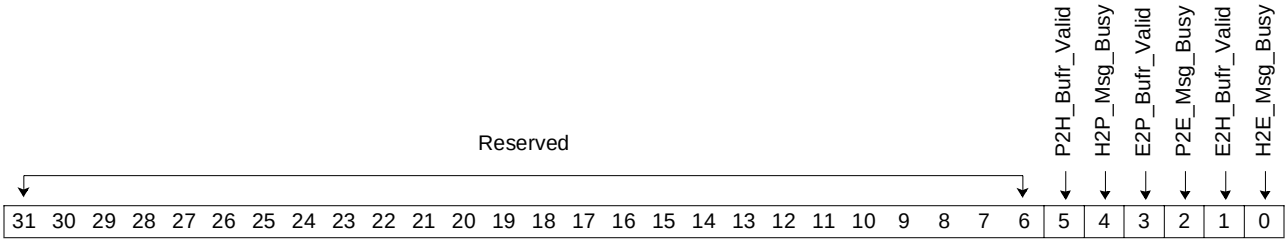
Field Name	PLB Bit	Reset	Description
H2E_Msg_Doorbell 31	0	0	PCI Host to EPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
H2E_Msg_Doorbell 30:1	1:30	0 for all	
H2E_Msg_Doorbell 0	31	0	

10.8.23 Message Status (Msg_Status) Register

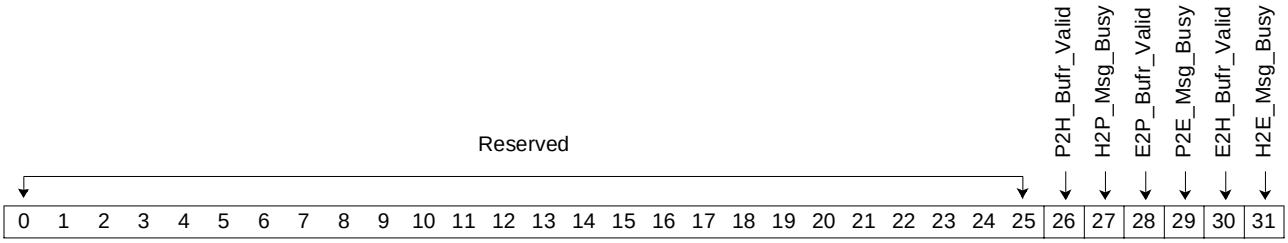
The Message Status register provides status information associated with inter-processor messaging. This read only register is accessible from either the PLB or the CAB for the purpose of checking status associated with messaging.

Access Type	CAB	Read
	PLB	Read
Base Address	CAB	x‘3801 4000’
	PLB	x‘7801 00A0’

CAB View



PLB View



CAB View

Field Name	CAB Bit(s)	Reset	Description
Reserved	31:6		Reserved
P2H_Bufr_Valid	5	0	PowerPC to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2P_Msg_Busy	4	0	PCI Host to PowerPC message busy indicator. 0 Not busy processing H2P message 1 Busy processing H2P message
E2P_Bufr_Valid	3	0	EPC to PowerPC message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
P2E_Msg_Busy	2	0	PowerPC to EPC message busy indicator. 0 Not busy processing P2E message 1 Busy processing P2E message
E2H_Bufr_Valid	1	0	EPC to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2E_Msg_Busy	0	0	PCI Host to EPC message busy indicator. 0 Not busy processing H2E message 1 Busy processing H2E message

PLB View

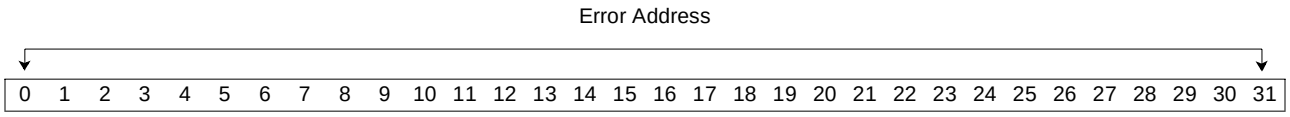
Field Name	Bit(s)	Reset	Description
Reserved	0:25		Reserved
P2H_Bufr_Valid	26	0	PowerPC to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2P_Msg_Busy	27	0	PCI Host to PowerPC message busy indicator. 0 Not busy processing H2P message 1 Busy processing H2P message
E2P_Bufr_Valid	28	0	EPC to PowerPC message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
P2E_Msg_Busy	29	0	PowerPC to EPC message busy indicator. 0 Not busy processing P2E message 1 Busy processing P2E message
E2H_Bufr_Valid	30	0	EPC to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2E_Msg_Busy	31	0	PCI Host to EPC message busy indicator. 0 Not busy processing H2E message 1 Busy processing H2E message

10.8.24 Slave Error Address Register (SEAR)

The Slave Error Address Register records the address value of the last occurrence of a parity error encountered by the DRAM Interface slave unit.

Access Type Read

Base Address (PLB) x‘7801 00B8’



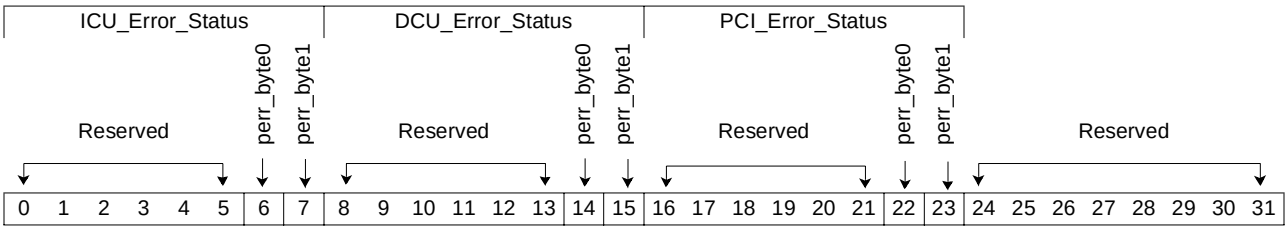
Field Name	Bit(s)	Reset	Description
Error Address	0:31	x’0000 0000’	Last PLB address value that resulted in a parity error when using the DRAM Interface slave unit.

10.8.25 Slave Error Status Register (SESR)

The Slave Error Status Register contains status information that indicates which masters have encountered parity errors in reading from the DRAM and whether these errors occurred in a byte with an even (Perr_Byte0) or an odd (Perr_Byte1) address value. The PowerPC subsystem has three PLB masters: the Instruction Cache Unit (ICU), the Data Cache Unit (DCU), and the PCI macro. The contents of this register are reset to x‘0000 0000’ when read.

Access Type Read and Reset

Base Address (PLB) x‘7801 00C0’



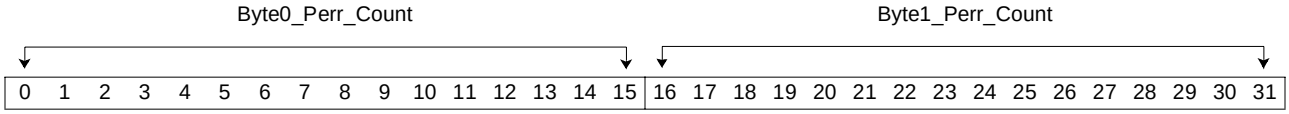
Field Name	Bit(s)	Reset	Description
Reserved	0:5		Reserved
Perr_Byte0	6	0	Byte 0 parity error indication - whether or not the Instruction Cache Unit PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Perr_Byte1	7	0	Byte 1 parity error indication- whether or not the Instruction Cache Unit PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Reserved	8:13		Reserved
Perr_Byte0	14	0	Byte 0 parity error indication - whether or not the Data Cache Unit PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Perr_Byte1	15	0	Byte 1 parity error indication - whether or not the Data Cache Unit PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Reserved	16:21		Reserved
Perr_Byte0	22	0	Byte 0 parity error indication - whether or not the PCI Macro PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Perr_Byte1	23	0	Byte 1 parity error indication - whether or not the PCI Macro PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Reserved	24:31		Reserved

10.8.26 Parity Error Counter (Perr_Count) Register

The Parity Error Counter register contains two 16-bit counter values. These counters accumulate the total number of parity errors for even and odd byte addresses since the last time the chip was reset. These counters rollover to zero when their maximum value is reached (65535).

Access Type Read

Base Address (PLB) x‘7801 00C8’



Field Name	Bit(s)	Reset	Description
Byte0_Perr_Count	0:15	x’0000’	Count of byte 0 parity errors encountered when reading DRAM from DRAM Interface slave unit.
Byte1_Perr_Count	16:31	x’0000’	Count of byte 1parity errors encountered when reading DRAM from DRAM Interface slave unit.

10.9 System Start-Up and Initialization

10.9.1 NP4GS3 Resets

Table 214: Reset Domains

Reset Domain	Applies
PowerPC Core	Applies only to the 405 processor core and is used to control its start-up during system initialization
Network Function	Applies to all other functions in the NP4GS3

A general reset is performed whenever the NP4GS3's Reset input pin is activated. A general reset includes the PowerPC Core, and the NP4GS3 Network Function reset domains. The PowerPC Core and the NP4GS3 Network Function reset domains are also controlled by separate configuration registers, PowerPC_Reset and Soft_Reset, respectively. Each of these registers are accessible via the NP4GS3's CAB Interface. A general reset sets the PowerPC_Reset to '1' (activated), but the Soft_Reset will be set to '0' (deactivated).

When the general reset is deactivated, the PowerPC Core remains in reset and the PowerPC Macros and the NP4GS3 Network Function are in an idle condition with state machines active and control structures un-initialized. The NP4GS3 Network Function will be functional once EPC picocode has been loaded and control structures are initialized. The EPC is activated by setting the Boot_Done control bit to '1'. The PowerPC Core is activated by clearing the bit of the PowerPC_Reset register.

The PowerPC Core and the NP4GS3 Network Function domains are separately reset and activated by setting and clearing their respective reset registers. Setting the control bit in the Soft_Reset register causes the Network Function to be momentarily reset while the PowerPC is held in reset. The PowerPC_Reset is also activated whenever the Watch Dog timer of the PowerPC Core expires for a second time and Watch Dog Reset Enable (WD_Reset_Ena) is set to '1'. When enabled, the second expiration of the Watch Dog timer results in a pulse on the SERR# signal of the PCI Bus. The TCR [WRC] of the PowerPC Core is set to '10' to generate a chip reset request on the second expiration of the Watch Dog timer.

The PowerPC Core can be reset from the RISCWatch debugger environment. A Core reset performed from RISCWatch resets the 405 Core, but does not set the PowerPC_Reset control register. This allows a momentary reset of the PowerPC Core and does not require a release of the PowerPC Core by clearing the PowerPC_Reset control register. The PowerPC Core is also reset and the PowerPC_Reset control register is set when a Chip reset is performed from RISCWatch. This resets the PowerPC Core and holds it in reset until the PowerPC_Reset register is cleared.

10.9.2 Systems Initialized by External PCI Host Processors

System implementations with control function centralized in a PCI Host processor are initialized primarily by an external PCI Host processor. These systems do not use the SPM Interface or its associated flash memory to boot the EPC and DASL picocode. The host processor loads code for the PowerPC processor and the EPC and DASL picoprocessors.

The general sequence of the start-up and initialization is as follows:

1. The host processor boots and performs blade level configuration and testing while holding blade subsystems in reset.
2. The host processor releases each of the blade subsystems' reset signals in sequence.
3. Release of the general reset input pin of the NP4GS3 activates its state machines. The NP4GS3 is in an idle state with un-initialized structures and the 405 Core remains in reset.
4. The host system uses the PCI Configuration protocol to configure internal registers of the PCI Interface Macro. This configuration sets the base address values of the PCI Target Maps. The PCI Master Maps are disabled.
5. The host processor uses the appropriate PCI target address values to configure and initialize the NP4GS3's DRAM controllers via the CAB Interface.
6. The host processor uses the appropriate PCI target address values to load the PowerPC code into one of the NP4GS3's DRAMs. This code includes the branch code loaded into the Boot_Redir_Inst registers.
7. The host loads the picocode memories of the EPC and DASL. These memories are accessible via the CAB Interface macro. The addresses of the registers controlling this interface were mapped to PCI addresses in step 4.
8. The host processor starts the 405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface. The PowerPC starts by fetching the instruction at PLB address x'FFFF FFFC'. This address is decoded by hardware to provide an unconditional branch to instruction space in the PowerPC's DRAM memory.
9. The host processor uses the CAB Interface to set the Boot_Done control bit to start the EPC code. This code controls the initialization of the NP4GS3 structures in preparation for network traffic. Alternatively, this initialization is performed by the host processor via the CAB Interface.
10. Communication ports are configured and enabled by either the host processor or the 405 Core.

10.9.3 Systems with PCI Host Processors and Initialized by PowerPC

The PowerPC primarily controls start-up and initialization sequences of systems of this category. These systems do not use the SPM Interface or its associated flash memory for booting the EPC boot picocode. The PowerPC loads code for the PowerPC and the EPC and DASL picoprocessors.

The general sequence of the start-up and initialization is as follows:

1. The host processor boots and performs blade level configuration and testing while holding blade subsystems in reset.
2. The host processor releases each of the blade subsystems' reset signals in sequence.
3. Release of the general reset input pin of the NP4GS3 activates its state machines. The NP4GS3 is in an idle state with un-initialized structures and the 405 Core remains in reset.
4. The host system uses the PCI Configuration protocol to configure internal registers of the PCI Interface Macro. This configuration sets the base address values of the PCI Target Maps. The PCI Master Maps are disabled except for the PMM0 which maps PLB addresses x'FFFE 0000' through x'FFFF FFFF' to the same addresses in PCI address space.
5. The host processor starts the 405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
6. The 405 Core will boot from code residing in the PCI address space starting at address x'FFFF FFFC'.
7. The 405 Core processor configures and initializes the NP4GS3's DRAM controllers via the CAB Interface.
8. The 405 Core processor loads the PowerPC code, via the DRAM Interface Macro, into one of the NP4GS3's DRAMs.
9. The 405 Core processor loads the picocode for the EPC and DASL via the CAB Interface.
10. Using the CAB Interface, the host processor sets the Boot_Done Control bit to start the EPC picocode. This picocode will control the initialization of the NP4GS3 structures in preparation for network traffic. Alternatively, this initialization is performed by the 405 Core processor via the CAB Interface.
11. Communication ports are configured and enabled by the 405 Core.

10.9.4 Systems Without PCI Host Processors and Initialized by PowerPC

The EPC initially controls start-up and initialization sequences of systems of this category, but they are primarily controlled by the PowerPC. These systems use the SPM Interface and its associated flash memory for booting the EPC picocode. The PowerPC loads code for the PowerPC and the EPC and DASL picoprocessors.

The general sequence of the start-up and initialization is as follows:

1. Release of the general reset input pin of the NP4GS3 activates its state machines. The PCI Master Maps are disabled except for the PMM0 which maps PLB addresses x‘FFFE 0000’ through x‘FFFF FFFF’ to the same addresses in PCI address space. EPC boot picocode is loaded from the flash memory via the SPM Interface hardware. The SPM Interface hardware sets the Boot_Done signal to start the EPC and the 405 Core remains in reset.
2. EPC code executes diagnostic and initialization code which includes the initialization of the NP4GS3’s DRAM controllers.
3. The EPC starts the 405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
4. The 405 Core will boot from code residing in the PCI address space starting at address x‘FFFF FFFC’.
5. The 405 Core processor loads the PowerPC code via the DRAM Interface Macro into one of the NP4GS3’s DRAMs.
6. The 405 Core processor or the EPC loads the picocode for the DASL via the CAB Interface.
7. Communication ports are configured and enabled by the 405 Core.

10.9.5 Systems Without PCI Interface Hardware and Initialized by EPC

The EPC controls start-up and initialization sequences of systems of this category. These systems use the SPM Interface and its associated flash memory for booting the EPC picocode. Code for the PowerPC and the EPC are loaded by the SPM Interface hardware and the EPC. Code for the PowerPC exists in the flash memory or is provided using Guided Traffic.

The general sequence of the start-up and initialization is as follows:

1. Release of the general reset input pin of the NP4GS3 activates its state machines. EPC picocode is loaded from the flash memory via the SPM Interface hardware. The Boot_Done signal is set by the SPM Interface hardware to start the EPC. The 405 Core remains in reset.
2. EPC code executes diagnostic and initialization code, which includes the initialization of the NP4GS3's DRAM controllers.
3. The EPC loads the code for the PowerPC from the flash memory into the DRAM. This code includes the branch code loaded into the Boot_Redir_Inst registers. Alternatively, this code is loaded using Guided Traffic. Guided Traffic flows once the communications port connecting the source has been enabled (see step 6.).
4. The EPC starts the 405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
5. The 405 Core will boot from code residing in the PLB address space starting at address x'FFFF FFFC'. This address is decoded by hardware to provide an unconditional branch to instruction space in the PowerPC's DRAM memory.
6. Communication ports are configured and enabled by the 405 Core or, alternatively, by the EPC.

11. Reset and Initialization

The intent of this section is to provide, by example, a method for initializing the NP4GS3. It is not intended to be exhaustive in scope, since there are many configurations and environments where the NP4GS3 may be applied. The external Serial Parallel Manager (SPM) Field Programmable Gate Array (FPGA) mentioned in this chapter is a component of the Network Processor Evaluation Kit. However, the design for the SPM FPGA is not provided by IBM.

11.1 Overview

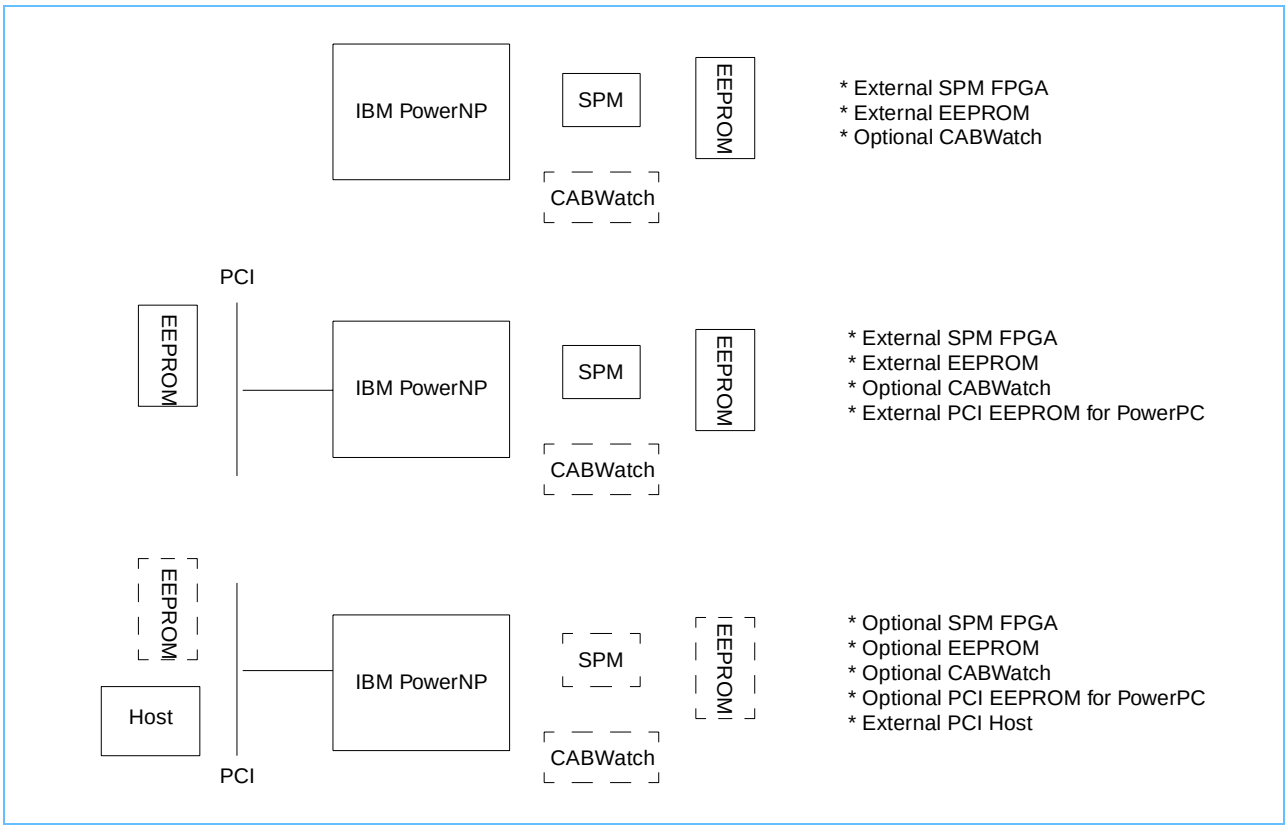
The NP4GS3 supports a variety of system and adapter configurations. In order to support a particular environment, the network processor must be initialized with parameters that match that environment's system or adapter requirements. The sequence of reset and initialization is shown in *Table 215*:

Table 215: Reset and Initialization Sequence

Step	Action	Notes
1	Set I/Os	Chip I/Os set to match adapter requirements
2	Reset	Must be held in reset for minimum of 1 µs
3	Boot	Several Boot options
4	Setup 1	Low Level Hardware setup
5	Diagnostics 1	Memory and Register Diagnostics
6	Setup 2	Basic Hardware setup
7	Hardware Initialization	Hardware self-initialization of various data structures
8	Diagnostics 2	Data Flow Diagnostics
9	Operational	Everything ready for first Guided Frame
10	Configure	Functional Configuration
11	Initialization Complete	Everything ready for first Data Frame

Some system environments do not require all of the steps and some require that certain steps be performed differently. The NP4GS3 supports the system environments shown in *Figure 73* for Reset and Initialization.

Figure 73: System Environments



The following sections describe each step in the Reset and Initialization sequence and the various ways to accomplish each step.

11.2 Step 1: Set I/Os

There are several NP4GS3 I/Os that must be set to appropriate values in order to operate correctly. Most of these I/Os will be set to a fixed value at the card level, but some will be set to the appropriate value based on system parameters. The following table lists all configurable I/Os that must be set prior to initial reset.

Table 216: Set I/Os Checklist

I/O	Values	Notes
Testmode(1:0)	See *Table 29: Miscellaneous Pins* on page 63 for the encoding of these I/O	Adapter may require mechanism to set test mode I/Os in order to force various test scenarios.
Boot_Picocode	See *Table 29: Miscellaneous Pins* on page 63 for the encoding of this I/O	Controls which interface loads the internal EPC instruction memory and boots the Guided Frame Handler thread.
Boot_PPC	See *Table 29: Miscellaneous Pins* on page 63 for the encoding of this I/O	Controls from where the internal PPC fetches its initial boot code.
PCI_Speed	See *Table 25: PCI Interface Pins* on page 59 for the encoding of this I/O	Controls the speed of the PCI bus
Switch_BNA	See *Table 29: Miscellaneous Pins* on page 63 for the encoding of this I/O	Initial value for Primary switch interface

Table 216: Set I/Os Checklist

I/O	Values	Notes
CPDetect_A	See *Table 14: PMM Interface Pin Multiplexing* on page 44 for the encoding of this I/O	Used to find a locally attached Control Point Function (CPF)
CPDetect_B	See *Table 14: PMM Interface Pin Multiplexing* on page 44 for the encoding of this I/O	Used to find a locally attached CPF
CPDetect_C	See *Table 14: PMM Interface Pin Multiplexing* on page 44 for the encoding of this I/O	Used to find a locally attached CPF
CPDetect_D	See *Table 14: PMM Interface Pin Multiplexing* on page 44 for the encoding of this I/O	Used to find a locally attached CPF
Spare_Tst_Rcvr(9:0)	See *Table 29: Miscellaneous Pins* on page 63 for the correct tie values for these I/O	Used during manufacturing testing

Additional configuration bits could be utilized by defining additional I/O on an external SPM FPGA as configuration inputs. If the user defines registers in the SPM CAB address space and a corresponding external SPM FPGA design, the NP4GS3 can read these SPM I/Os and make configuration adjustments based on their values (for example, the type of CP interface might be 100 Mb or 1 Gb). Custom boot picocode is required to take advantage of these additional features.

All clocks must be operating prior to issuing a reset to the NP4GS3. The Clock_Core, Switch Clock A, and Switch Clock B drive internal PLLs that must lock before the internal clock logic will release the Operational signal. The Clock125 and DMU_*(3) inputs, if applicable, should also be operating in order to properly reset the DMU interfaces. The PCI Clock (PCI_Clk) must be operating in order to properly reset the PCI Interface.

11.3 Step 2: Reset the NP4GS3

Once all the configuration I/Os are set and the clocks are running, the NP4GS3 can be reset using the Blade_Reset signal. This signal must be held active for a minimum of 1 µs and then returned to its inactive state. Internal logic requires an additional 101 µs to allow the PLLs to lock and all internal logic to be reset. The PCI Bus must be idle during this interval to ensure proper initialization of the PCI bus state machine. At this point, the NP4GS3 is ready to be booted. In addition to the Blade_Reset signal, the NP4GS3 supports two other “soft” reset mechanisms: The Soft_Reset register allows the entire NP4GS3 to be reset (just like a Blade_Reset), and the PowerPC_Reset register allows the PowerPC core to be reset. A Blade_Reset or Soft_Reset causes the PowerPC_Reset register to activate and hold the PowerPC Core in reset until this register is cleared. Therefore, a Blade_Reset resets the entire NP4GS3 and holds the PowerPC Core in reset until it is released by the EPC or an external host using the PCI bus.

11.4 Step 3: Boot

11.4.1 Boot the Embedded Processor Complex (EPC)

Booting the NP4GS3's EPC involves loading the internal picocode instruction space and turning over control of execution to the GFH thread. The GFH thread executes the loaded picocode and completes the appropriate steps in the bringup process. The NP4GS3 supports four ways to load the internal picocode instruction space: through the SPM Interface Logic, through the PCI bus from an external host, through the embedded PowerPC, or through CABWatch. Once the picocode instruction space is loaded, the Boot Done signal needs to be set by the loading device using the Boot_Override register, and the GFH thread starts executing the code stored at address '0'.

11.4.2 Boot the PowerPC

There are two steps in the process of booting the NP4GS3's embedded PowerPC. First, using the Boot_PPC I/O, the PowerPC support logic must be configured to boot the PowerPC from either the external D6 DRAM or the PCI bus. Second, the PowerPC must be released to execute the appropriate boot code. The PowerPC boot code can be mapped either into PCI address space or into the NP4GS3's external D6 DRAM, depending on the setting of the Boot_PPC I/O.

If an external Host processor is used on the PCI bus, it should use the PCI configuration protocol to set the NP4GS3's PCI Target Maps for access into the network processor's internal address space.

If the Boot_PPC I/O chooses the PCI bus, the internal PLB bus addresses x'FFFE 0000' through x'FFFF FFFF' are mapped to PCI address space. Once the PowerPC_Reset register is cleared (by either the EPC or by an external host across the PCI bus), the PowerPC will fetch and execute the boot code from across the PCI bus.

If the Boot_PPC I/O chooses the external D6 DRAM, the D6 DRAM must be written with the appropriate boot code and the Boot_Redir_Inst registers must be written to point to the code in the D6 DRAM before the PowerPC_Reset register is released. The internal logic maps the PowerPC's boot addresses of x'FFFF FFE0' through x'FFFF FFFC' to the Boot_Redir_Inst registers and the remaining boot code is fetched from the external D6 DRAM. The D6 DRAM and the Boot_Redir_Inst registers can be written by either the EPC or an external host on the PCI bus. When everything is set up, use a CAB write to clear the PowerPC_Reset register to allow the PowerPC Core to execute the boot code.

11.4.3 Boot Summary

The EPC must be booted by first loading its picocode instructions (by either the SPM, an external PCI host, the Embedded PowerPC, or CABWatch) and then issuing the Boot Done signal (by the picocode loader). If the Embedded PowerPC is to be used, it must have its instruction space loaded (if D6 is used), then pointing the boot logic to the appropriate boot location (PCI or D6), and finally releasing the PowerPC_Reset register (by either the EPC, an external PCI host, or CABWatch). Once one or both systems are booted, the following steps can be performed by one or both processing complexes. (Some accesses to external memories can only be performed by the EPC complex.)

11.5 Step 4: Setup 1

Setup 1 is needed to set some low level hardware functions that enable the NP4GS3 to interface with its external DRAMs and to configure some internal registers that enable the execution of Step 5: diagnostics 1. Setup 1 should configure or check the following registers according to the system setup and usage:

Table 217: Setup 1 Checklist

Register	Fields	Notes
Memory Configuration Register	All	This register is set to match the populated external memories.
DRAM Parameter Register	All	This register is set to match the external DRAM's characteristics.
Thread Enable Register	All	This register enables the non-GFH threads. The GFH is always enabled.
Initialization Register	DRAM Cntl Start	This bit starts the DRAM interfaces.
Initialization Done Register	CM Init Done E_DS Init Done	These bits indicate that the DRAM initialization has completed.

11.6 Step 5: Diagnostics 1

Diagnostics 1 tests internal registers, internal memories, and external memories as required by the diagnostics program (read and write tests). This step comes before the hardware initialization step because several of these structures will be initialized by the hardware to contain functional data structures. By testing these structures first, the diagnostics program does not need to be concerned with corrupting the contents of these locations during hardware initialization. Care must be taken that the values written to the structures do not force an undesirable situation (such as soft resetting the chip). However, most of these structures can be tested by the diagnostics program to ensure proper operation. *Table 218* lists some of the structures that could be tested by this step.

Table 218: Diagnostics 1 Checklist

Structure	Test	Notes
Phase Locked Loop Fail	Verify all PLLs locked	If any PLL fails, any further operation is questionable
DPPU Processors	ALU, Scratch Memory, Internal Processor Registers	Test each thread
Ingress Data Store	Read/Write	
Egress Data Store	Read/Write	
Control Store D0-D4	Read/Write	
Control Store D6	Read/Write	Coordinated with PowerPC code loading
Control Store H0-H1	Read/Write	
Counter Definition Table	Configure	Setup to test Counters
Counter Memory	Read/Write/Add	
Policy Manager Memory	Read/Write	
Egress QCBs	Read/Write	
Egress RCB	Read/Write	
Egress Target Port Queues	Read/Write	
MCCA	Read/Write	
PMM Rx/Tx Counters	Read/Write	
PMM SA Tables	Read/Write	
Ingress BCB	Read/Write	
Ingress FCB	Read/Write	Not all fields are testable
CIA Memory	Read/Write	
Ingress Flow Control Probability Memory	Read/Write	
Egress Flow Control Probability Memory	Read/Write	
DASL-A Picocode Memory	Read/Write	
DASL-B Picocode Memory	Read/Write	
Various Internal Configuration Registers	Read/Write	Not all fields will be testable, care must be taken when changing certain control bits.

11.7 Step 6: Setup 2

Setup 2 is needed to setup the hardware for self-initialization and to configure the hardware structures for operational state. These configuration registers must be set to the desirable values based on the system design and usage:

Table 219: Setup 2 Checklist

Register	Fields	Notes
Master Grant Mode	All	
TB Mode	All	
Egress Reassembly Sequence Check	All	
Aborted Frame Reassembly Action Control	All	
Packing Control	All	
Ingress BCB_FQ Thresholds	All	
Egress SDM Stack Threshold	All	
Free Queue Extended Stack Maximum Size	All	
Egress FQ Thresholds	All	
G Queue Maximum Size	All	
DMU Configuration	All	DMU Configuration can be postponed until Step 10: Configure if DMU Init Start is also postponed. DMU for CPF must be done during Setup 2. If the DMU is configured, the appropriate external Physical Devices must also be configured.
Packet Over SONET Control	All	POS Configuration can be postponed until Step 10: Configure if DMU Init Start is also postponed.
Ethernet Encapsulation Type for Control	All	
Ethernet Encapsulation Type for Data	All	
DASL Picocode Memory	Both A and B	Written with appropriate DASL picocode.
DASL Initialization and Configuration	Primary Set	DASL Init can be postponed until the Configure Step if DASL Start is also postponed and CPF is locally attached.
DASL Initialization and Configuration	DASL Wrap Mode	DASL Wrap Mode can be postponed until the Configure Step.
DASL Wrap	All	

11.8 Step 7: Hardware Initialization

Hardware Initialization allows the NP4GS3 to self-initialize several internal structures, thereby decreasing the overall time required to prepare the processor for operation. Several internal structures will be initialized with free lists, default values, or initial states in order to accept the first guided frames from the CPF. Once these data structures are initialized, the picocode should not modify them with further read/write diagnostics. To initiate the hardware self-initialization, the registers shown in *Table 220* need to be written.

Table 220: Hardware Initialization Checklist

Register	Fields	Notes
DASL Start	All	Only start the DASL interface after the Primary Set of DASL configuration bits have been configured.
DASL Initialization and Configuration	Alt_Ena	Only start the Alternate DASL after the Primary DASL has been started.
Initialization	DMU set	Only start each DMU after its associated DMU Configuration has been set.
Initialization	Functional Island	Starts all other island's self-initialization.

11.9 Step 8: Diagnostics 2

Diagnostics 2 determines if the NP4GS3 is ready for operation and allows testing of data flow paths. The items listed in *Table 221* should be setup, checked, and/or tested.

Table 221: Diagnostic 2 Checklist

Register	Fields	Notes
Initialization Done	All started in Hardware Initialization Step	The code polls this register until a timeout occurs or all expected bits are set. DASL timeout = 20 ms E_EDS timeout = 15 ms
DASL Initialization and Configuration	Pri_Sync_Term	If Primary DASL was initialized and Sync Termination should occur from the network processor, this register should be set to cause IDLE cells to be sent.
DASL Initialization and Configuration	Alt_Sync_Term	If Alternate DASL was initialized and Sync Termination should occur from the network processor, this register should be set to cause IDLE cells to be sent.
LU Def Table	Read/Write Test	These structures can only be tested after Hardware Initialization.
SMT Compare Table	Read/Write Test	These structures can only be tested after Hardware Initialization.
Tree Search Free Queues	Read/Write Test	These structures can only be tested after Hardware Initialization. Not all fields are testable
Port Configuration Table	All	Set to default values for Diagnostics 2
LU Def Table	All	Set to default values for Diagnostics 2
Ingress Target Port Data Storage Map	All	Set to default values for Diagnostics 2
Target Port Data Storage Map	All	Set to default values for Diagnostics 2
Build Frame on Egress Side		Lease twins and store test frame in Egress Data Store
Half Wrap		Wrap test frame from Egress to Ingress using Wrap DMU
Full Wrap		Wrap Ingress frame back to Egress side if DASL in wrap mode or DASL has been completely configured including Target Blade information.
External Wrap		If external Physical Devices are configured, full external wraps can be performed.
Tree Searches		To test the tree search logic, tree searches can be performed on a pre-built sample tree written to memory.

11.10 Step 9: Operational

After the Diagnostics 2 tests have finished, any previously written default values may need to be updated to allow this step to proceed. If all Diagnostics have passed, the Operational signal can be activated to indicate to an external CPF that the NP4GS3 is ready to receive Guided Frames. This signal is activated by writing the NP4GS3 Ready register which then activates the $\overline{\text{Operational}}$ I/O. If some portion of the diagnostics have not passed, the Ready register should not be written. This causes the CPF to timeout and realize the diagnostic failure. To determine what portion of the diagnostics have failed, the system designer must make provisions at the board or system level to record the status in a location that is accessible by the CPF. One method is to provide an I^2C interface to an external SPM FPGA which the CPF could access.

11.11 Step 10: Configure

After the Operational signal has been activated, the CPF can send Guided Frames to the NP4GS3 for functional configuration. Items that can be configured include:

Table 222: Configure Checklist

Register	Fields	Notes
DASL Initialization and Configuration	Primary Set	The Primary Set can be configured if postponed during the Setup 2 Step.
DASL Initialization and Configuration	DASL Wrap Mode	DASL Wrap Mode can be set if postponed during Setup 2 Step.
DASL Start	All	Primary DASL can be started if postponed during Setup 2 Step.
DASL Initialization and Configuration	Alt_Ena	The Alternate Set can be configured if postponed during the Setup 2 Step.
Initialization Done	P_DASL Init Done	This bit should be polled if started during the Configure Step
Initialization Done	A_DASL Init Done	This bit should be polled if started during the Configure Step
DASL Initialization and Configuration	Pri_Sync_Term	If Primary DASL was initialized and Sync Termination should occur from the Network Processor, this register should be set to cause IDLE cells to be sent.
DASL Initialization and Configuration	Alt_Sync_Term	If Alternate DASL was initialized and Sync Termination should occur from the Network Processor, this register should be set to cause IDLE cells to be sent.
DMU Configuration	All	Configure now if DMU Configuration was postponed during Setup 2 Step. If the DMU is configured, the appropriate external Physical Devices also need to be configured.
Packet Over SONET Control	All	Configure now if POS Configuration was postponed during Setup 2 Step.
Functional Picocode	All	Functional Picocode should be loaded into the Instruction Memory
Port Config Table	All	Functional values for the PCT should be set
LU Def Table	All	Functional values should be set.
CIA Memory	All	Functional values should be set.
Hardware Classifier E_Type	All	Functional values should be set.
Hardware Classifier SAP	All	Functional values should be set.
Hardware Classifier PPP_Type	All	Functional values should be set.

Table 222: Configure Checklist (Continued)

Register	Fields	Notes
Interrupt Masks	All	Functional values should be set.
Timer Target	All	Functional values should be set.
Interrupt Target	All	Functional values should be set.
Address Bounds Check Control	All	Functional values should be set.
Static Table Entries	All	Any Static Table Entries should be loaded.
Ingress Target Port Data Storage Map	All	
My Target Blade Address	All	
Local Target Blade Vector	All	
Local MC Target Blade Vector	All	
Target Port Data Storage Map	All	
Egress QCBs	All	
QD Accuracy	All	
SA Address Array	All	
CPF Address	All	
Counter Definition Table	All	
Counters	All	Any Counters used by Counter Manager must be Read/ Cleared.
Policy Manager Memory	All	Setup initial values for Policies.

11.12 Step 11: Initialization Complete

Once steps 1 through 10 are complete, and all the items on the checklists have been configured, the NP4GS3 is ready for data traffic. The Ports can be enabled (at the Physical Devices) and Switch Cells can start to flow.

12. Debug Facilities

12.1 Debugging Picoprocessors

The NP4GS3 provides several mechanisms to facilitate debugging of the picoprocessors.

12.1.1 Single Step

Each thread of the NP4GS3 can be enabled individually for single step instruction execution. Single step is defined as advancing the instruction address by one cycle and executing the instruction accordingly for enabled threads. Coprocessors are not affected by single step mode. Therefore coprocessor operations that at “live” speed would take several cycles may seem to take only one cycle in single step mode.

There are two ways to enable a thread for single step operation. The first is to write the Single Step Enable Register. This register is a single step bit mask for each thread and can be accessed through the Control Access Bus (CAB). The second is the Single Step Exception Register. This register is also a bit mask, one for each thread, but when set indicates which threads are to be placed into single step mode on a class3 interrupt . When a thread is in Single Step Mode, the thread can only be advanced by writing the Single Step Command register.

12.1.2 Break Points

The NP4GS3 supports one instruction break point that is shared by all of the threads. When a thread’s instruction address matches the break point address, a class3 level interrupt is generated. This causes all threads enabled in the Single Step Exception Register to enter single step mode. The break point address is configured in the Break Point Address Register.

12.1.3 CAB Accessible Registers

The scalar and array registers of the Core Language Processor (CLP) and of the DPPU coprocessors are accessible through the CAB for evaluation purposes. The CLP’s General Purpose registers, which are directly accessible with the CLP, are mapped to read only scalar registers on the Control Access Bus.

12.2 RISCWatch

The NP4GS3 supports RISCWatch through the JTAG interface.

RISCWatch is a hardware and software development tool for the PowerPC 600 Family of microprocessors and the PowerPC 400Series of Embedded Controllers. The source-level debugger and processor-control features provide developers with the tools needed to develop and debug hardware and software quickly and efficiently.

Developers who take advantage of RISCWatch are provided a wealth of advanced debug capabilities. Among the advanced features of this full-functioned debugger are: real-time trace, ethernet hardware interface, C++ support, extensive command file support, and on-chip debug support. In the future, multi-processing will be supported. All this in a debugger that supports both XCOFF and the Embedded ABI for PowerPC industry standard.

RISCWatch includes:

- On-chip debugging via IEEE 1149.1 (JTAG) interface
- Target monitor debugging
- OS Open Real-Time Operating System aware debugging
- Source-level and assembler debugging of C/C++ executables
- Real-time trace support via the RISCTrace feature for the PowerPC 400Series
- Network support for remote debugging of the system under development
- Supports industry standard Embedded ABI for PowerPC and XCOFF ABI
- Command-file support for automated test and command sequences
- Simple and reliable 16-pin interface to the system under development
- Ethernet to target JTAG interface hardware
- Multiple hosts supported
- Intuitive and easy-to-use windowed user interface that reduces development time

For more information, go to http://www.chips.ibm.com/products/powerpc/tools/riscwatc.html

.

13. IBM PowerNP Configuration

The NP4GS3 must be configured after internal diagnostics have run. Configuration is performed by a CPF that generates guided traffic to write configuration registers. These configuration registers are reset by the hardware to a minimal option set.

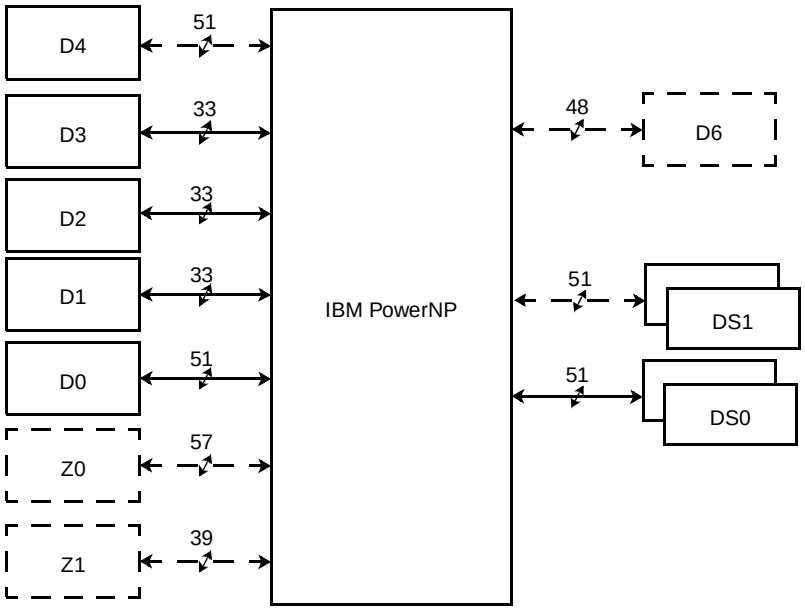
The following sections describe all configuration registers and their reset state. A base address and offset is provided.

13.1 Memory Configuration

The NP4GS3 is supported by a number of memory subsystems as shown in Figure 2 below. These memories contain data buffers and controls used by the NP4GS3. The D0, D1, D2, D3, D4, DS_0, and DS_1 are required for the base configuration of the NP4GS3 to operate. The other memory subsystems, Z0, Z1, and D6, are optional and provide additional functionality or capability when added to the required set of memory subsystems.

In its base configuration, the NP4GS3 does not perform enhanced scheduling, and has a limited look-up search capability. The enabling of memory subsystem interfaces is controlled by the contents of the Memory Configuration Register. The bits in this register are set by hardware during reset to enable the base configuration.

Figure 74: NP4GS3 Memory Subsystems

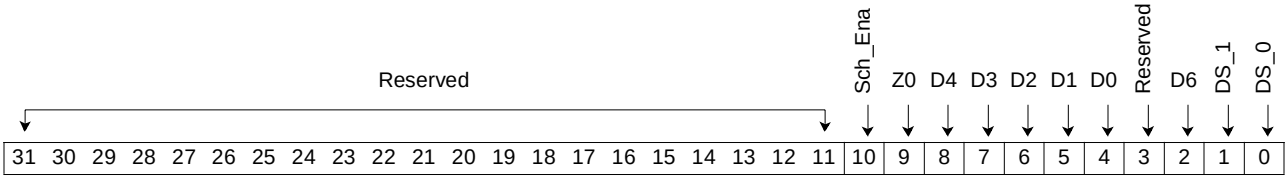


13.1.1 Memory Configuration Register (Memory_Config)

The memory configuration register enables or disables memory interfaces. It also enables the egress scheduler and the Z1 memory interface required by the egress scheduler to operate.

Access Type Read/Write

Base Address x‘A000 0120’



Field Name	Bit(s)	Reset	Description
Reserved	31:11		Reserved
Sch_Ena	10	0	Scheduler Enabled control enables both egress scheduler and Z1 memory interface. 1 Scheduler enabled 0 Scheduler disabled When setting this to a value of 1, the target port queue count Qcnt_PQ must be zero for all ports. When setting this value to 0, the target port queue count Qcnt_PQ+FQ must be zero for all ports.
Z0	9	0	Z0 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
D4	8	1	D4 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
D3	7	1	D3 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
D2	6	1	D2 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
D1	5	1	D1 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
D0	4	1	D0 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
Reserved	3	0	Reserved
D6	2	0	D6 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled
DS_1	1	1	DS_1 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled

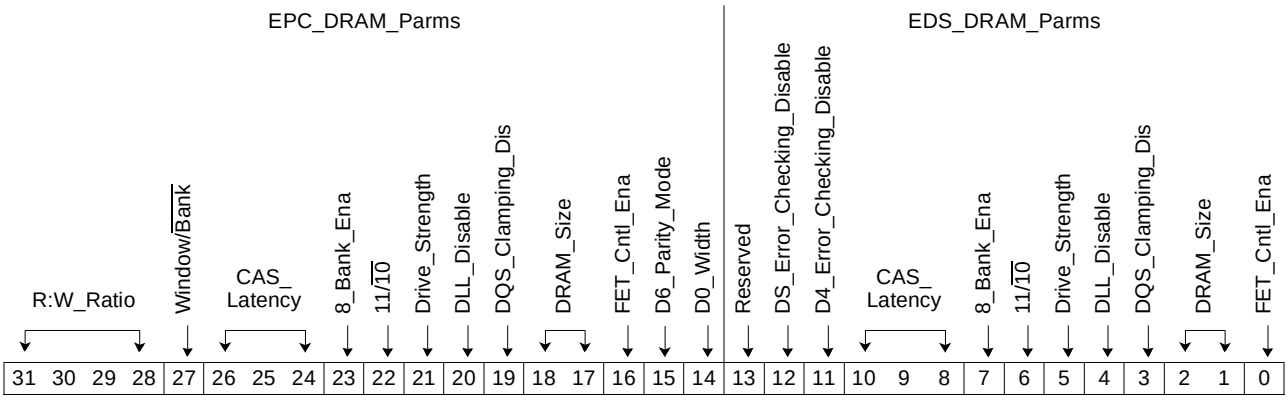
Field Name	Bit(s)	Reset	Description
DS_0	0	1	DS_0 Memory Subsystem Interface Enabled control 1 Interface enabled 0 Interface disabled

13.1.2 DRAM Parameter Register (DRAM_Parm)

The DRAM Parameter register controls the operation of the DRAMs used by the EPC and the Egress EDS. These DRAMs are controlled separately.

Access Type Read/Write

Base Address x‘A000 2400’



Field Name	Bit(s)	Reset	Description
R:W_Ratio	31:28	x‘F’	Read to Write Ratio controls the ratio of read to write window accesses associated with the EPC DRAMs.
Window/$\overline{\text{Bank}}$	27	0	Window or Bank control value controls access to the EPC DRAMs on a window by window or bank by bank basis. 0 Bank by bank access 1 Window by window access
CAS_Latency	26:24, 10:8	x‘010’	DRAM Column Address Strobe Latency value corresponds to the DRAM’s read latency measured from the Column Address. 000 - 001 Reserved 010 2 clock cycles 011 3 clock cycles 100 - 101 Reserved 110 2.5 clock cycles 111 Reserved
8_Bank_Enable	23, 7	0	Eight Bank Addressing Mode Enable control value.
11/$\overline{10}$	22, 6	1	Eleven or ten cycle DRAM control value controls the number of core clock cycles the DRAM controller uses to define an access window. 0 10 cycle DRAM 1 11 cycle DRAM
Drive_Strength	21, 5	0	DRAM Drive Strength control
DLL_Disable	20, 4	0	DLL Disable control
DQS_Clamping_Dis	19, 3	0	DQS Disable control
DRAM_Size	18:17	00	DRAM Size indicates the size of the Control Stores D0-D3 in DRAMs. 00 4x1Mx16 DDR DRAM, Burst=4 01 4x2Mx16 DDR DRAM, Burst=4 10 4x4Mx16 DDR DRAM, Burst=4 11 Reserved
FET_Cntl_Ena	16, 0	0	FET Control Enable control

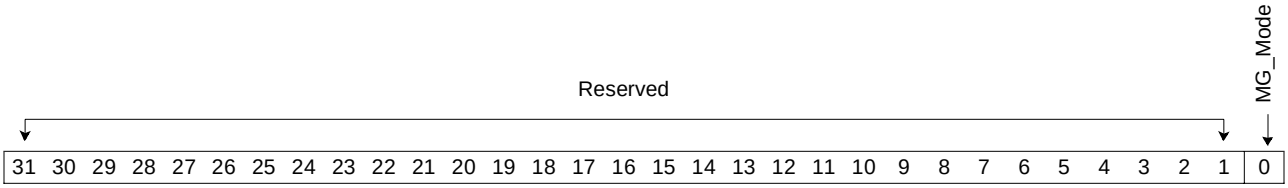
Field Name	Bit(s)	Reset	Description
D6_Parity_Mode	15	1	DRAM D6 Parity Mode Disable control value. 0 D6 interface supports an additional 2 DDR DRAMs which support byte parity. The hardware generates on write and checks parity on read. 1 D6 interface does not support parity.
D0_Width	14	1	D0 Width control indicates if one or two DRAMs are used for the D0 CS. 0 Single wide configuration using one DRAM. A single bank access provides 64 bits of data. 1 Double wide configuration using two DRAMs. A single bank access provides 128 bits of data
Reserved	13		Reserved
DS_Error_Checking_Disable	12	0	Egress Datastore Error Checking Disable control. When this field is set to 1, all DRAM error checking for the egress datastore is disabled.
D4_Error_Checking_Disable	11	0	D4 DRAM Error Checking Disable. When this field is set to 1, all DRAM error checking for the D4 DRAM is disabled.
DRAM_Size	2:1	00	DRAM Size indicates the size of the Egress Datastore and D4 DRAMs. 00 4x1Mx16 DDR DRAM, Burst=4, x2 01 4x2Mx16 DDR DRAM, Burst=4, x2 10 4x4Mx16 DDR DRAM, Burst=4, x2 11 Reserved The setting of this field affects the size of the extended stack queues as follows: GQ 00 01 10 11 GTQ 48 K 96 K 192 K Reserved PPQ 48 K 96 K 192 K Reserved GFQ 96 K 192 K 384 K Reserved GR0 96 K 192 K 384 K Reserved GR1 96 K 192 K 384 K Reserved GB0 96 K 192 K 384 K Reserved GB1 96 K 192 K 384 K Reserved Discard 96 K 192 K 384 K Reserved

13.2 Master Grant Mode Register (MG_Mode)

This configuration register configures the Master Grant IO (MGrant_A, MGrant_B) for either nested priority or independent priority mode.

Access Type Read/Write

Base Address x'A000 0820'



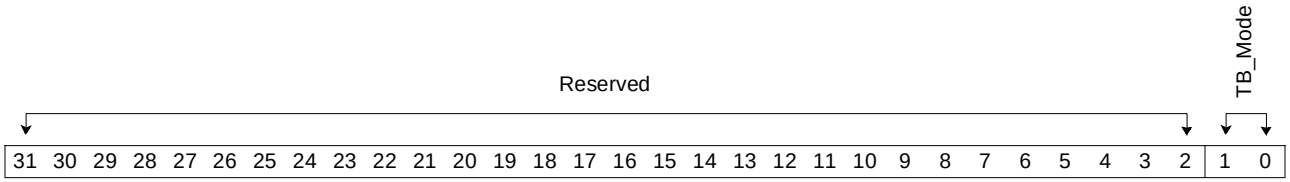
Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
MG_Mode	0	0	0 The MGrant IO is defined for nested priority encoding which is defined as: 00 No grant (on any priority) 01 Priority 0 has grant 10 Priority 0 and 1 have grant 11 Priority 0, 1 and 2 have grant 1 The MGrant IO is defined for independent priority encoding which is defined as: 00 No grant (on any priority) 01 Priority 0 and 1 have grant 10 Priority 2 has grant 11 Priority 0, 1 and 2 have grant

13.3 TB Mode Register (TB_Mode)

The target blade mode configures the maximum number of target network processors supported.

Access Type Read/Write

Base Address x'A000 0410'



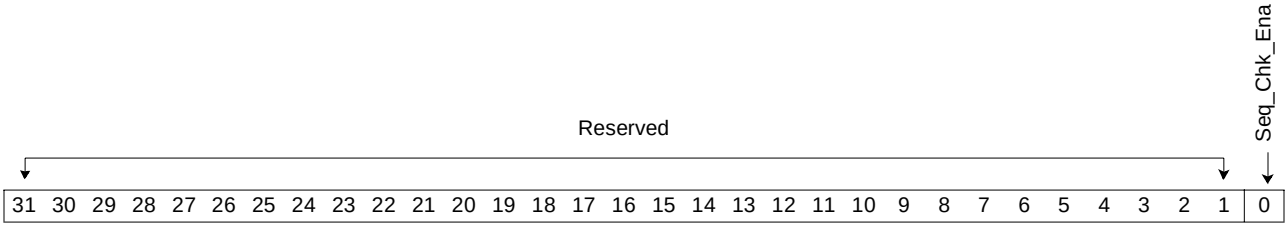
Field Name	Bit(s)	Reset	Description
Reserved	31:2		Reserved
TB_Mode	1:0	00	Target Blade Mode. This field is used to define the target blade mode currently in use by the NPR. The field is defined as: 00 16 blade mode. Valid addresses are 0:15. Multicast is indicated as a 16-bit vector. 01 Reserved 10 64 blade mode. Valid unicast target blade field encodes are 0 through 63. Multicast encodes are in the range of 512 through 65535. 11 Reserved

13.4 Egress Reassembly Sequence Check Register (E_Reassembly_Seq_Ck)

This configuration register enables sequence checking by the egress reassembly logic. The sequence checking insures that start of frame, optional middle of frame, and end of frame indications occur in the expected order for each cell of a frame being reassembled. Each cell that does not indicate start or end of frame carries a sequence number that is checked for proper order.

Access Type Read/Write

Base Address x'A000 0420'



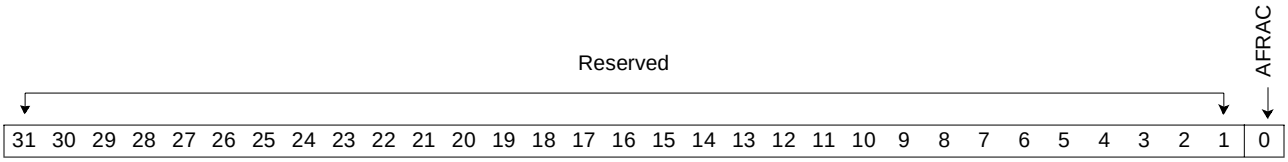
Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
Seq_Chk_Ena	0	1	Sequence Check Enable control 0 Sequence checking disabled 1 Sequence checking enabled for the E_EDS

13.5 Aborted Frame Reassembly Action Control Register (AFRAC)

This configuration register controls the action the hardware takes on a frame whose reassembly was aborted due to the receipt of an abort command in a cell header for the frame.

Access Type Read/Write

Base Address x'A000 0440'



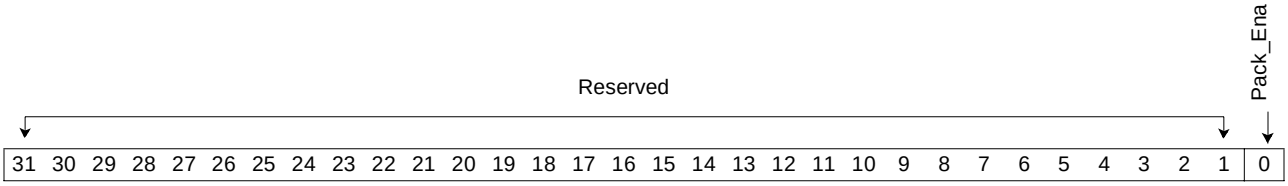
Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
AFRAC	0	0	Aborted Frame Reassembly Action Control 0 Aborted frames are enqueued to the Discard Queue 1 Aborted frames are enqueued with other frames on the associated GQ

13.6 Packing Control Register (Pack_Ctrl)

This configuration register is used to enable cell packing by the I-SDM.

Access Type Read/Write

Base Address x'A000 0480'



Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
Pack_Ena	0	1	Packing Enabled flag 0 Cell packing disabled 1 Cell packing enabled

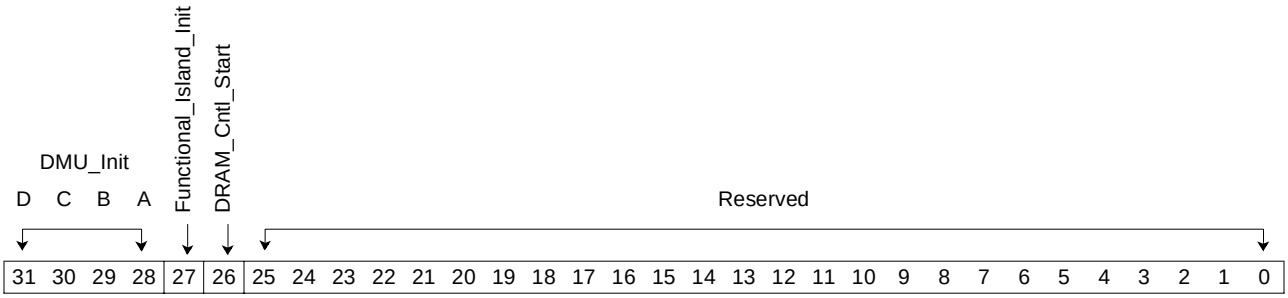
13.7 Initialization Control Registers

13.7.1 Initialization Register (Init)

This register controls the initialization of the functional islands. Each functional island and the DRAM Controllers begin initialization when the bits in this register are set to '1'. Each functional island signals the successful completion of initialization by setting its bit in the Init_Done Register. Once a functional island has been initialized, changing the state of these bits will not longer have any effect until a reset occurs.

Access Type Read/Write

Base Address x'A000 8100'



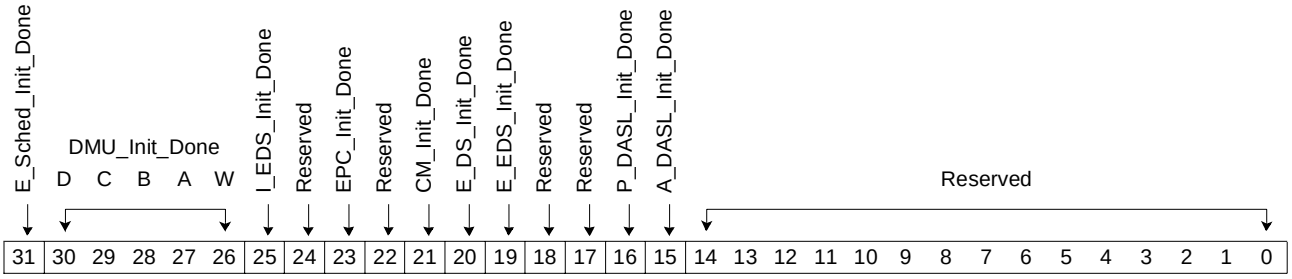
Field Name	Bit(s)	Reset	Description
DMU_Init	31:28	0000	Data Mover Unit Initialization control individually initializes each DMU.
Functional_Island_Init	27	0	Functional Island Initialization control. 0 Nop 1 Functional islands start hardware initialization. Completion of hardware initialization is reported in the Initialization Done Register.
DRAM_Cntl_Start	26	0	DRAM Controller Start Control. 0 Nop 1 Causes the DRAM Controllers to initialize and start operation. When initialization completes, the DRAM controllers set the Cntl_Mem Init Done and E-DS Init Done bits of the Initialization Done Register.
Reserved	25:0		Reserved

13.7.2 Initialization Done Register (Init_Done)

This register indicates that functional islands have completed their initialization when the corresponding bits are set to '1'. This register tracks the initialization state of the functional islands. The GCH reads this register during the initialization of the NP4GS3.

Access Type Read Only

Base Address x'A000 8200'

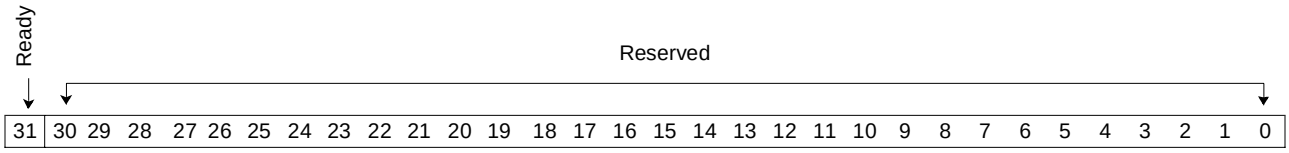


Field Name	Bit(s)	Reset	Description
E_Sched_Init_Done	31	0	Set to '1' by the hardware when the egress scheduler hardware completes its initialization.
DMU_Init_Done	30:26	0	Set to '1' by the hardware when the DMU hardware has completed its initialization. Bit DMU 30 D 29 C 28 B 27 A 26 Wrap
I_EDS_Init_Done	25	0	Set to '1' by the hardware when the Ingress EDS completes its initialization.
Reserved	24		Reserved
EPC_Init_Done	23	0	Set to '1' by the hardware when the EPC completes its initialization.
Reserved	22		Reserved
CM_Init_Done	21	0	Set to '1' by the hardware when the TSE's DRAM controller for the TSE completes its initialization.
E_DS_Init_Done	20	0	Set to '1' by the hardware when the Egress Data Stores' DRAM controller completes its initialization.
E_EDS_Init_Done	19	0	Set to '1' by the hardware when the Egress EDS completes its initialization
Reserved	18:17		Reserved
P_DASL_Init_Done	16	0	Set to '1' by the hardware when the primary DASL completes its initialization. The DASL interface continues to send synchronization cells.
A_DASL_Init_Done	15	0	Set to '1' by the hardware when the alternate DASL completes its initialization. The DASL interface continues to sent synchronization cells.
Reserved	14:0		Reserved

13.8 Network Processor Ready Register (NPR_Ready)

Access Type Read/Write

Base Address x‘A004 0020’

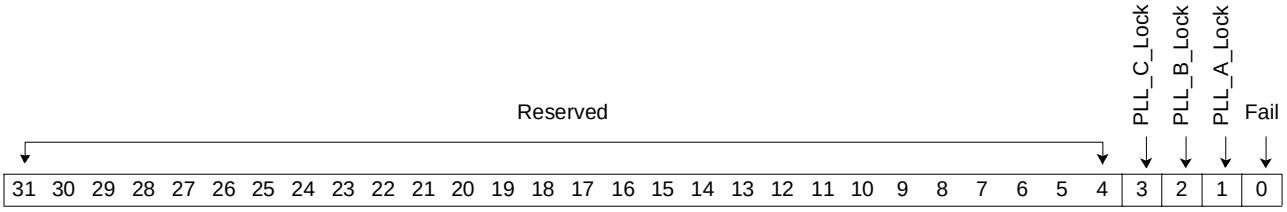


Field Name	Bit(s)	Reset	Description
Ready	31	0	Ready is configured by picocode to drive a chip pin to indicate that the NP4GS3 has been initialized and is ready to receive Guided Traffic. 0 NP4GS3 not ready 1 NP4GS3 ready
Reserved	30:0		Reserved

13.9 Phase Locked Loop Fail Register (PLL_Lock_Fail)

Access Type Read Only

Base Address x‘A000 0220’



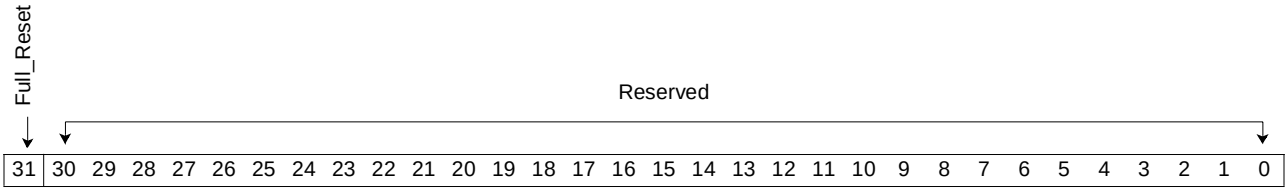
Field Name	Bit(s)	Reset	Description
Reserved	31:4		Reserved
PLL_C_Lock	3		Current status of lock indictor of the Core Clock PLL 0 Phase/frequency lock 1 Phase/frequency seek
PLL_B_Lock	2		Current status of lock indictor of the DASL-B PLL 0 Phase/frequency lock 1 Phase/frequency seek
PLL_A_Lock	1		Current status of lock indictor of the DASL-A PLL 0 Phase/frequency lock 1 Phase/frequency seek
Fail	0		Phased Locked Loop Fail indicator - indicates that an on-chip PLL has failed. This field is written by the clock logic. 0 PLLs OK 1 PLL failed If Fail is indicated at the end of the reset interval (101 micro seconds after reset is started) the operational chip IO is set to ‘1’. After the end of the reset interval, a change in Fail will not affect operational chip IO.

13.10 Software Controlled Reset Register (Soft_Reset)

This register provides a control for software to reset the network processor.

Access Type Write Only

Base Address x‘A000 0240’



Field Name	Bit(s)	Reset	Description
Full_Reset	31	0	Full Reset value resets the NP4GS3 hardware via the picocode. This is the same full reset function provided by the Blade_Reset IO. 0 Reserved 1 NP4GS3 performs an internal hardware reset
Reserved	30:0		Reserved

13.11 Ingress Free Queue Threshold Configuration

13.11.1 BCB_FQ Threshold Registers

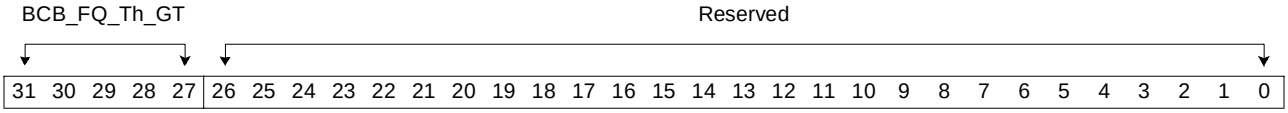
The value of the queue count in the BCB_FQ Control Block is continuously compared to the values each of the three threshold registers contains. The result of this comparison affects the NP4GS3's flow control mechanisms. The values in these registers must be chosen such that BCB_FQ_Th_GT ≤ BCB_FQ_Th_0 ≤ BCB_FQ_Th_1 ≤ BCB_FQ_Th_2. For proper operation the minimum value for BCB_FQ_Th_GT is x'08'.

13.11.2 BCB_FQ Threshold for Guided Traffic (BCB_FQ_Th_GT)

The Ingress EDS reads this register to determine when to discard all packets received.

Access Type Read/Write

Base Address x'A000 1080'



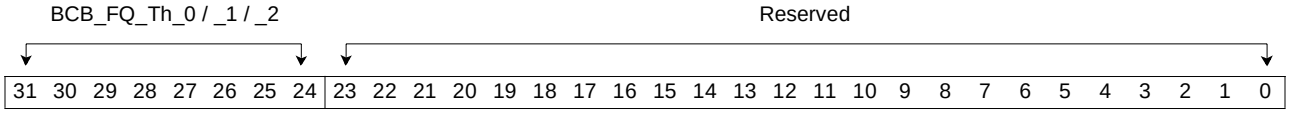
Field Name	Bit(s)	Reset	Description
BCB_FQ_Th_GT	31:27	x'08'	BCB Free Queue Threshold GT value is measured in units of individual buffers. For example, a threshold value of x'01' represents a threshold of one buffer.
Reserved	26:0		Reserved

13.11.3 BCB_FQ_Threshold_0 / _1 / _2 Registers (BCB_FQ_Th_0/_1/_2)

Access Type Read/Write

Base Address

BCB_FQ_Th_0	x‘A000 1010’
BCB_FQ_Th_1	x‘A000 1020’
BCB_FQ_Th_2	x‘A000 1040’



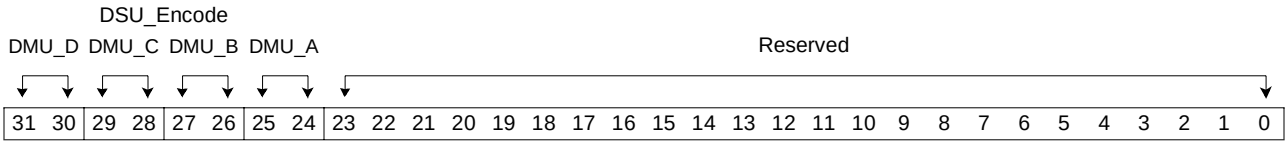
Field Name	Bit(s)	Reset	Description
BCB_FQ_Th_0 BCB_FQ_Th_1 BCB_FQ_Th_2	31:24	x‘00’	BCB Free Queue Threshold 0 / 1 / 2 value, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. The Ingress EDS reads this field to determine when to perform a discard action. Violation of this threshold (BCB queue count is less than this threshold), sends an interrupt to the EPC. When BCB_FQ_Th_0 is violated, the discard action is to perform partial packet discards. New data buffers are not allocated for frame traffic and portions of frames may be lost. When BCB_FQ_Th_1 is violated, the discard action is to perform packet discards. New data buffers are not allocated for new frame traffic, but frames already started continue to allocated buffers as needed.
Reserved	23:0		Reserved

13.12 Ingress Target DMU Data Storage Map Register (I_TDMU_DSU)

This register defines the Egress Data Storage Units (DSU) used for each DMU.

Access Type Read/Write

Base Address x‘A000 0180’



Field Name	Bit(s)	Reset	Description
DSU_Encode	31:24	00	During enqueue operations, the values in this configuration register are loaded into the ingress Frame Control Block’s DSU field when a DSU value is not specified by the enqueue. The hardware determines the target DMU from enqueue information and uses the corresponding field in this configuration register to load the DSU field. Four fields are defined, one for each DMU, with the following encode: 00 DSU 0 01 DSU 1 10 Reserved 11 DSU0, DSU1 Bits DMU 31:30 DMU_D 29:28 DMU_C 27:26 DMU_B 24:25 DMU_A
Reserved	23:0		Reserved

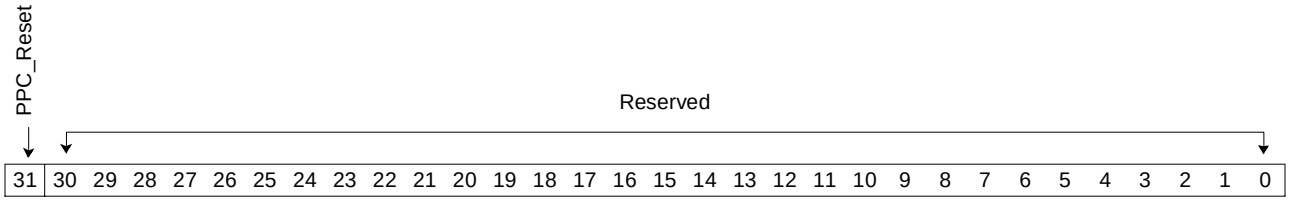
13.13 Embedded Processor Complex Configuration

13.13.1 PowerPC Core Reset Register (PowerPC_Reset)

This register contains a control value used to hold the PowerPC core in reset state.

Access Type Read/Write

Base Address x‘A000 8010’



Field Name	Bit(s)	Reset	Description
PPC_Reset	31	1	PowerPC Core Reset - Holds the Power PC core in a reset state when set to 1. The rest of the Power PC functional island is not affected by this control and can only be reset by a full reset. 0 PowerPC core reset disabled 1 PowerPC core held in reset
Reserved	30:0		Reserved

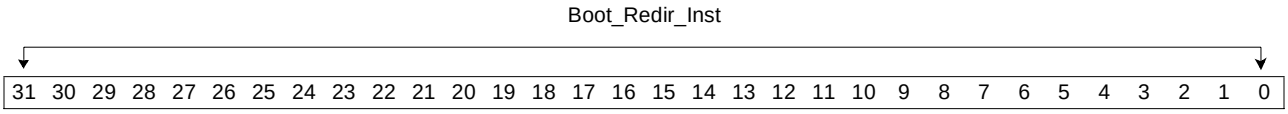
13.13.2 PowerPC Boot Redirection Instruction Registers (Boot_Redir_Inst)

In system implementations in which the embedded PowerPC boots from the D6 DRAM, the Mailbox and DRAM Interface macro performs PowerPC boot address redirection. Under these conditions, the hardware provides instructions that redirect the boot sequence to a location in the PowerPC's DRAM. Storage for eight instructions is provided by the Boot_Redir_Inst registers.

The PowerPC Boot Redirection Instruction (Boot_Redir_Inst) registers are accessed from the CAB Interface. These registers provide capacity for eight instructions for PLB addresses x'FFFFFFE0' - x'FFFFFFFC' These instructions redirect the PowerPC to boot from a location in the PowerPC's DRAM and are configured before the PPC_Reset is cleared.

Access Type Read/Write

Base Address x'3800 0110' - x'3800 0117'



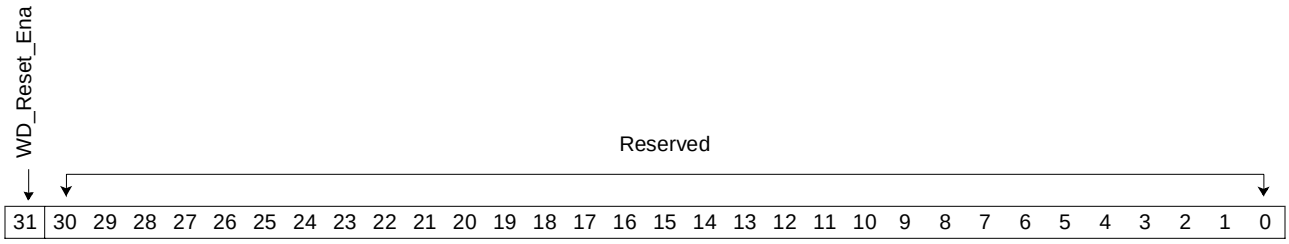
Field Name	Bit(s)	Reset	Description
Boot_Redir_Inst	31:0	x'0000 0000'	PowerPC boot redirection instruction address values contains instructions used by the PowerPC to redirect the boot sequence to a location in D6 DRAM. Offset — Corresponding PowerPC Instruction Address 0 — x'FFFF FFE0' 1 — x'FFFF FFE4' 2 — x'FFFF FFE8' 3 — x'FFFF FFEC' 4 — x'FFFF FFF0' 5 — x'FFFF FFF4' 6 — x'FFFF FFF8' 7 — x'FFFF FFFC'

13.13.3 Watch Dog Reset Enable Register (WD_Reset_Ena)

This register controls the action of a Watch Dog timer expiration. When set to 1, the second expiration of the Watch Dog timer causes a reset to the PowerPC core.

Access Type Read/Write

Base Address x‘A000 4800’



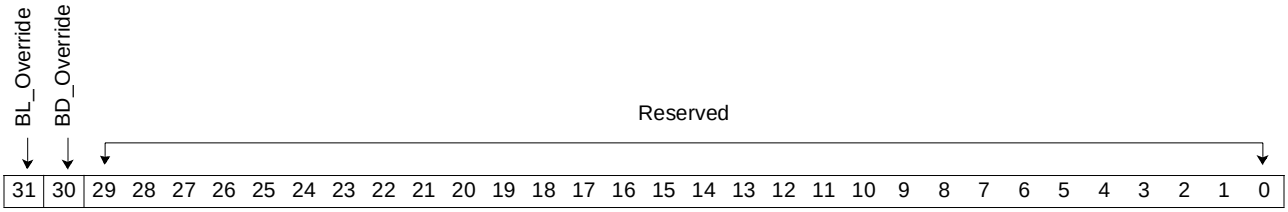
Field Name	Bit(s)	Reset	Description
WD_Reset_Ena	31	0	Reset Enable. 0 Disable reset of PowerPC core 1 Enable reset of PowerPC core on Watch Dog expire
Reserved	30:0		Reserved

13.13.4 Boot Override Register (Boot_Override)

This register provides boot sequence control in a debug environment. When the NP4GS3 is powered on in debug mode, the BL_Override bit is set and the SPM interface is inhibited from loading the boot picocode. Using CABwatch, different boot picocode can be loaded via the CABwatch interface. The GFH can then be started by setting the BD_Override bit to '1'. Alternatively, the BL_Override bit can be set to '0', and the boot picocode will be loaded by the SPM interface.

Access Type Read/Write

Base Address x'A000 8800'

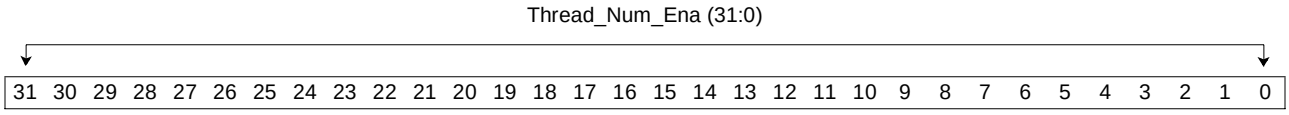


Field Name	Bit(s)	Reset	Description
BL_Override	31	See description	This is set to the value of the boot_picocode IO during reset. 0 Boot code is loaded by the SPM interface state machine. 1 Boot code is loaded by intervention of software. The bootcode can be loaded using the CABWatch interface, using the PCI bus, or by using the embedded Power PC. When reset to '1', a CAB write can set this field to '0' to start the SPM interface controlled boot sequence. The SPM interface reads this field to control the behavior of its state machine.
BD_Override	30	0	Boot Done Override control value. 0 Nop 1 When the SPM interface controlled boot loading sequence is overridden, this bit is set after the EPC's Instruction Memory has been loaded to start the EPC. A CAB write can set this field to '1' to indicate that the EPC's Instruction Memory is loaded. The Configuration Register logic reads this field when generating the POR Interrupt to the EPC.
Reserved	29:0		Reserved

13.13.5 Thread Enable Register (Thread_Enable)

This register contains control information used to enable or disable each thread.

Access Type	Read/Write	Bits 31:1
	Read Only	Bit 0
Base Address	x‘A000 8020’	



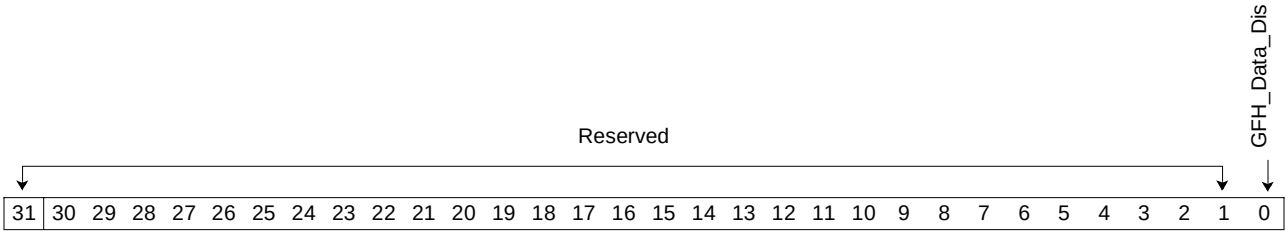
Field Name	Bit(s)	Reset	Description
Thread_Num_Ena (31:1)	31:1	0	Thread enable. 0 Disabled 1 Corresponding thread enabled for use
Thread_Num_Ena 0	0	1	Thread 0 (the GFH) is always enabled and cannot be disabled through this bit.

13.13.6 GFH Data Disable Register (GFH_Data_Dis)

This register is used to enable the Dispatch to assign data frames to the GFH for processing.

Access Type Read/Write

Base Address x‘24C0 0030’



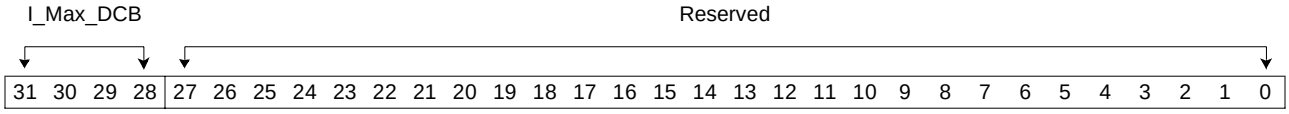
Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
GFH_Data_Dis	0	0	Guided Frame Handler Data Enable control. 0 Enabled 1 Not Enabled This field is configured to enable or disable the dispatching of data frames to the GFH.

13.13.7 Ingress Maximum DCB Entries (I_Max_DCB)

This register defines the maximum number of ingress frames that are currently allowed to be simultaneously serviced by the Dispatch unit. This limits the total number of ingress frames occupying space in the dispatcher control block.

Access Type Read/Write

Base Address x‘2440 0C40’



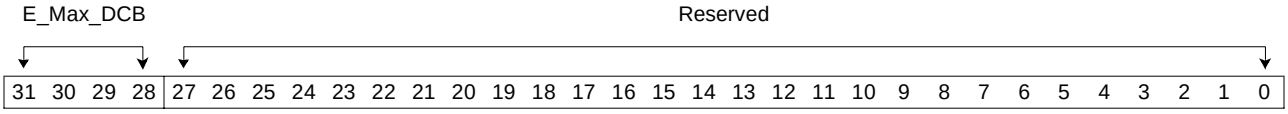
Field Name	Bit(s)	Reset	Description
I_Max_DCB	31:28	x‘6’	Maximum number of ingress frames allowed service in the dispatcher control block.
Reserved	27:0		Reserved

13.13.8 Egress Maximum DCB Entries (E_Max_DCB)

This register defines the maximum number of egress frames that are currently allowed to be simultaneously serviced by the Dispatch unit. This limits the total number of egress frames occupying space in the dispatcher control block.

Access Type Read/Write

Base Address x‘2440 0C50’



Field Name	Bit(s)	Reset	Description
E_Max_DCB	31:28	x‘6’	Maximum number of egress frames allowed service in the dispatcher control block.
Reserved	27:0		Reserved

13.13.9 My Target Blade Address Register (My_TB)

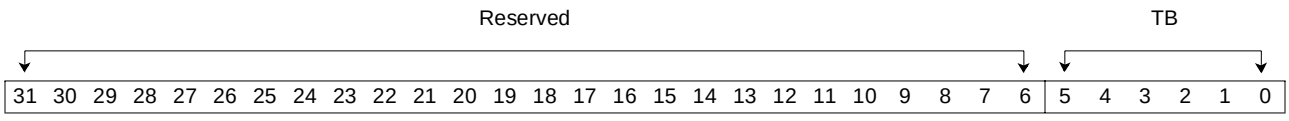
This register contains the local blade address value.

Access Type	Master Copy	Read/Write
	Thread Copies	Read Only

Base Addresses

Thread	Address
Master Copy	x‘A000 4080’
0	x‘2000 0890’
1	x‘2010 0890’
2	x‘2020 0890’
3	x‘2030 0890’
4	x‘2040 0890’
5	x‘2050 0890’
6	x‘2060 0890’
7	x‘2070 0890’
8	x‘2080 0890’
9	x‘2090 0890’
10	x‘20A0 0890’
11	x‘20B0 0890’
12	x‘20C0 0890’
13	x‘20D0 0890’
14	x‘20E0 0890’
15	x‘20F0 0890’

Thread	Address
16	x‘2100 0890’
17	x‘2110 0890’
18	x‘2120 0890’
19	x‘2130 0890’
20	x‘2140 0890’
21	x‘2150 0890’
22	x‘2160 0890’
23	x‘2170 0890’
24	x‘2180 0890’
25	x‘2190 0890’
26	x‘21A0 0890’
27	x‘21B0 0890’
28	x‘21C0 0890’
29	x‘21D0 0890’
30	x‘21E0 0890’
31	x‘21F0 0890’



Field Name	Bit(s)	Reset	Description
Reserved	31:6		Reserved
TB	5:0	0	Blade address of this network processor. The value in this field is limited by the TB_Mode configured (See the IBM PowerNP NP4GS3 Hardware Reference Manual, Section 1). It is further limited when configured for DASL Wrap mode (See the IBM PowerNP NP4GS3 Hardware Reference Manual, Section 1) to a value of either 1 or 0.

13.13.10 Local Target Blade Vector Register (Local_TB_Vector)

When both the DASL-A and DASL-B are active, this register is used to determine the interface used when forwarding traffic. The register is defined as a target blade bit vector where each bit in the register represents a target blade address.

For unicast traffic, in all target blade modes, the target blade address (defined in both the FCBPage and FCB2) is used to select the appropriate bit in the register for comparison. If the selected bit is set to 1, then local DASL interface is used to transmit the cell, otherwise the remote DASL interface is used

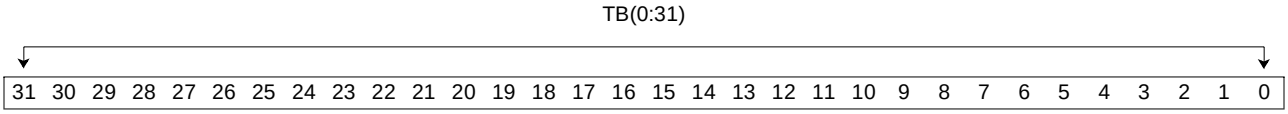
For multicast traffic (in 16 blade mode only) the target blade address, (defined in both the FCBPage and FCB2 as a bit vector) is compared bit by bit to the contents of this register. When a bit in the target blade address is set to 1, and the corresponding bit in this register is also set to 1, then the local DASL interface is used to transmit the cell. When a bit in the target blade address is set to 1 and the corresponding bit in this register is set to 0, then the remote DASL interface is used to transmit the cell.

For multicast traffic (in 64 blade mode) the Local MCTarget Blade Vector Register is used (see *Local MCTarget Blade Vector Register (Local_MC_TB_Max)* on page 367).

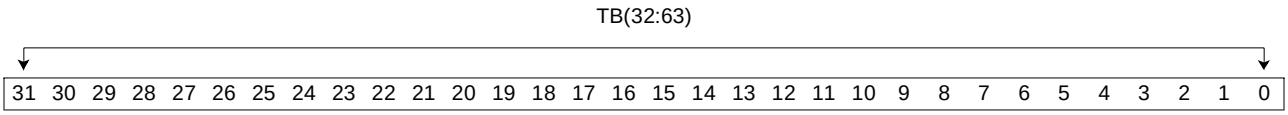
Access Type Read/Write

Base Addresses x‘A000 4100’

Base Address Offset 0



Base Address Offset 1



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
TB(0:31)	31:0	0	This is a bit vector representing target blade addresses 0 to 31.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
TB(32:63)	31:0	0	This is a bit vector representing target blade addresses 32 to 63.

13.13.11 Local MCTarget Blade Vector Register (Local_MC_TB_Max)

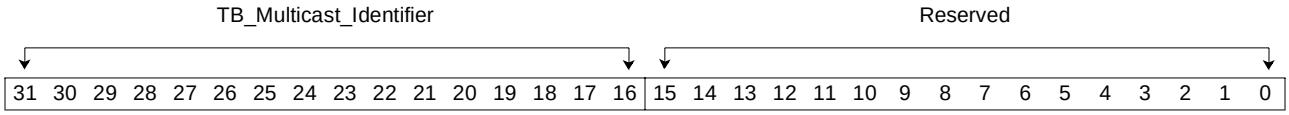
When configured for 64 blade mode, with both the DASL-A and the DASL-B active, this register is used to determine the interface to be used when forwarding multicast traffic.

The target blade address (defined in both the FCBPage and FCB2) is compared to the value in the TB_Multicast_Indentifier field. If the target blade address is less than this value, then the local DASL interface is used to transmit the cell, otherwise the remote DASL interface is used.

The TB_Multicast_Indentifier field is reset to 0, causing all multicast traffic to use the remote DASL, during power on.

Access Type Read/Write

Base Addresses x'A000 4200'



Field Name	Bit(s)	Reset	Description
TB_Multicast_Identifier	31:16	0	Multicast local maximum. The TB field for a frame is compared to the value of this field; when smaller the frame is local. Used only when not configured for 16 blade mode.
Reserved	15:0		Reserved

13.14 Flow Control Structures

13.14.1 Ingress Flow Control Hardware Structures

13.14.1.1 Ingress Transmit Probability Memory Register (I_Tx_Prob_Mem)

The Ingress Flow Control Hardware contains an internal memory that holds 64 different transmit probabilities for Flow Control.

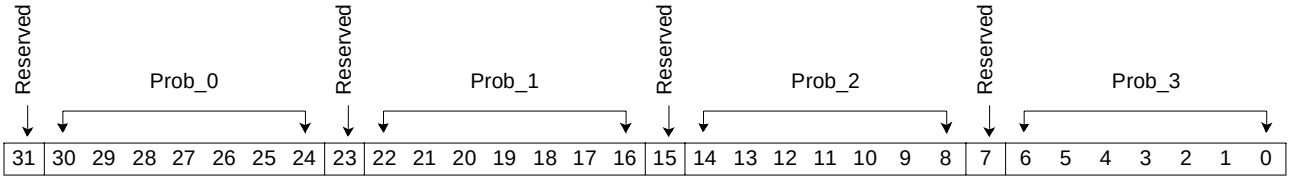
The probability memory occupies 16 entries in the CAB address space. Each probability entry in the CAB is 32 bits wide and contains four 7-bit probabilities. The 4-bit access address to each probability entry comprises two components: the 3-bit QosClass field (QQQ) and the 1-bit Remote Egress Status Bus value for the current priority (T). The address is formed as QQQT. The QosClass is taken from the Ingress FCBPage. The Remote Egress Status Bus value for the current priority reflects the threshold status of the Egress' leased twin count.

The Ingress Flow Control Hardware accesses probabilities within each probability memory entry by using the 2-bit Color portion of the FC_Info field taken from the Ingress FCBPage as an index. The probabilities are organized as shown below.

Access Type Read/Write

Base Address x'3000 00#0'

Note: 'The base address is listed with a '#' replacing one of the hex digits. The '#' ranges from x'0' to x'F', and indicates which probability entry is being referenced.



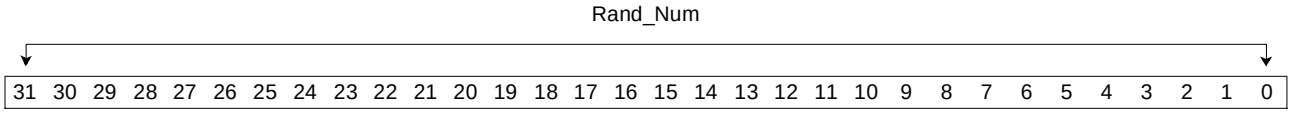
Field Name	Bit(s)	Reset	Description
Reserved	31		Reserved
Prob_0	30:24		Transmit Probability 0 - transmit probability accessed when Color is '11'
Reserved	23		Reserved
Prob_1	22:16		Transmit Probability 1 - transmit probability accessed when Color is '10'
Reserved	15		Reserved
Prob_2	14:8		Transmit Probability 2- transmit probability accessed when Color is '01'
Reserved	7		Reserved
Prob_3	6:0		Transmit Probability 3- transmit probability accessed when Color is '00'

13.14.1.2 Ingress Pseudo-Random Number Register (I_Rand_Num)

This register contains a 32-bit pseudo-random number used in the Flow Control algorithms. The CAB accesses this register in order to modify its starting point in the pseudo-random sequence. However, a write to this register is not necessary to start the pseudo-random sequence: it starts generating pseudo-random numbers as soon as the reset is finished.

Access Type Read/Write

Base Addresses x‘3000 0100’



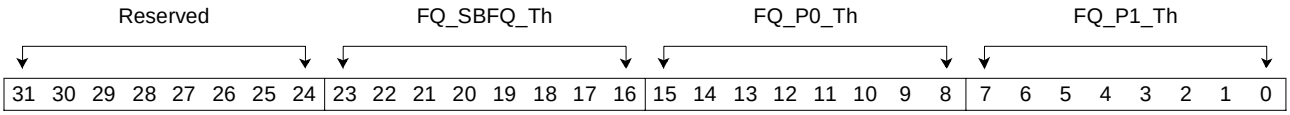
Field Name	Bit(s)	Reset	Description
Rand_Num	31:0		32-bit pseudo-random number

13.14.1.3 Free Queue Thresholds Register (FQ_Th)

This register contains three thresholds that are compared against the Ingress Free Queue. The results of this comparison are used in the Flow Control algorithms. Thresholds are in units of 16 buffers.

Access Type Read/Write

Base Addresses x‘A040 0020’



Field Name	Bit(s)	Reset	Description
Reserved	31:24		Reserved
FQ_SBFQ_Th	23:16	x‘FF’	Source Blade Free Queue Threshold value compared against the Ingress Free Queue Count and used to set the I_FreeQ_Th IO to '1'.
FQ_P0_Th	15:8	x‘FF’	Source Blade Free Queue Threshold value for Priority 0 traffic that is compared against the Ingress Free Queue Count.
FQ_P1_Th	7:0	x‘FF’	Source Blade Free Queue Threshold value for Priority 1 traffic that is compared against the Ingress Free Queue Count.

13.14.2 Egress Flow Control Structures

13.14.2.1 Egress Transmit Probability Memory (E_Tx_Prob_Mem) Register

The Egress Flow Control Hardware contains an internal memory that holds 64 different transmit probabilities for Flow Control.

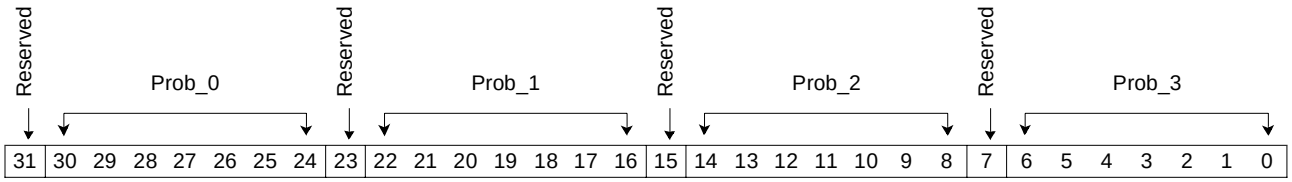
The probability memory occupies 16 entries in the CAB address space. Each probability entry in the CAB is 32 bits wide and contains four 7-bit probabilities. An entry in the Egress Probability Memory is accessed by using the 4-bit FC_Info field taken from the Egress FCBPage as an index.

The Egress Flow Control Hardware uses a 2-bit address to access the probabilities within each probability memory entry. This address (formed as FP) comprises two components: a 1-bit “threshold exceeded” value for the current priority of the flow queue count , and a 1-bit “threshold exceeded” value for the current priority of the combined flow/port queue count .

Access Type Read/Write

Base Addresses x‘B000 00#0’

Note: The base address is listed with a ‘#’ replacing one of the hex digits. The ‘#’ ranges from x‘0’ to x‘F’, and indicates which probability entry is being referenced.



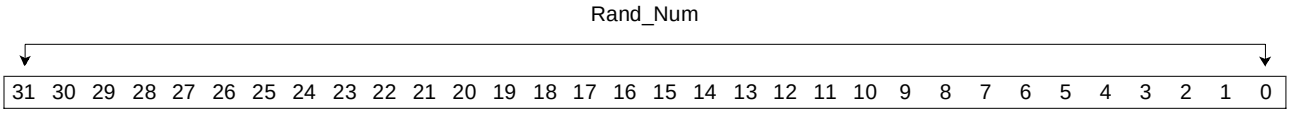
Field Name	Bit(s)	Reset	Description
Reserved	31		Reserved
Prob_0	30:24		Transmit Probability 0 - transmit probability accessed when ‘FP’ is ‘11’
Reserved	23		Reserved
Prob_1	22:16		Transmit Probability 1 - transmit probability accessed when ‘FP’ is ‘10’
Reserved	15		Reserved
Prob_2	14:8		Transmit Probability 2- transmit probability accessed when ‘FP’ is ‘01’
Reserved	7		Reserved
Prob_3	6:0		Transmit Probability 3- transmit probability accessed when ‘FP’ is ‘00’

13.14.2.2 Egress Pseudo-Random Number (E_Rand_Num)

This register contains a 32-bit pseudo-random number used in the Flow Control algorithms. The CAB accesses this register in order to modify its starting point in the pseudo-random sequence. However, a write to this register is not necessary to start the pseudo-random sequence: it starts generating pseudo-random numbers as soon as the reset is finished.

Access Type Read/Write

Base Addresses x‘B000 0100’



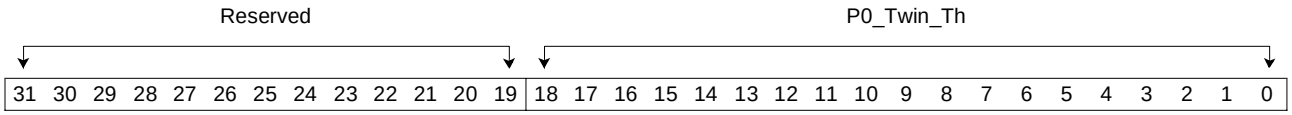
Field Name	Bit(s)	Reset	Description
Rand_Num	31:0		32-bit pseudo-random number

13.14.2.3 P0 Twin Count Threshold (P0_Twin_Th)

This register contains the threshold value that is compared against the Priority 0 Twin Count. The results of this comparison are used in the Flow Control algorithms.

Access Type Read/Write

Base Addresses x‘A040 0100’



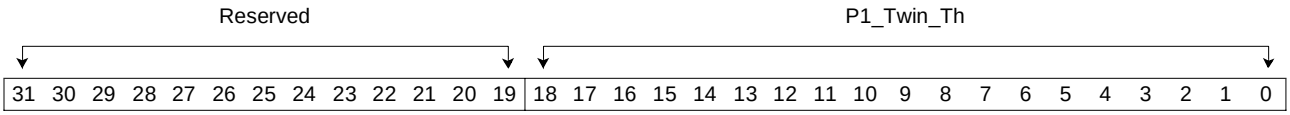
Field Name	Bit(s)	Reset	Description
Reserved	31:19		Reserved
P0_Twin_Th	18:0	x‘0 0000’	P0 Twin Count Threshold value used in the Egress Flow Control Hardware algorithm.

13.14.2.4 P1 Twin Count Threshold (P1_Twin_Th)

This register contains the threshold value that is compared against the Priority 1 Twin Count. The results of this comparison are used in the Flow Control algorithms.

Access Type Read/Write

Base Addresses x‘A040 0200’



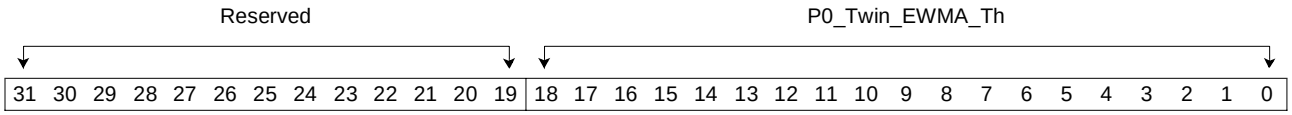
Field Name	Bit(s)	Reset	Description
Reserved	31:19		Reserved
P1_Twin_Th	18:0	x‘0 0000’	P1 Twin Count Threshold value used in the Egress Flow Control Hardware algorithm.

13.14.2.5 Egress P0 Twin Count EWMA Threshold Register (E_P0_Twin_EWMA_Th)

This register contains the threshold value that is compared against the Egress P0 Twin Count EWMA . The results of this comparison are placed on the Remote Egress Status Bus.

Access Type Read/Write

Base Addresses x‘A040 0400’



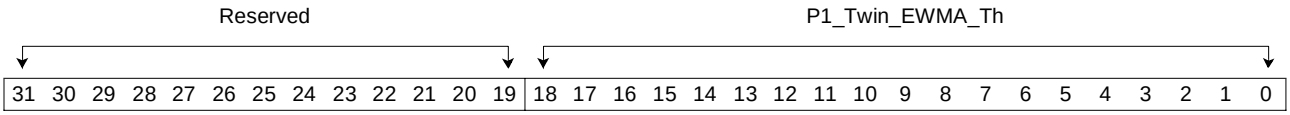
Field Name	Bit(s)	Reset	Description
Reserved	31:19		Reserved
P0_Twin_EWMA_Th	18:0	x‘0 0000’	Priority 0 Egress Leased Twin Count Exponentially Weighted Moving Average Threshold value. This value is compared against the Egress P0 Twin Count EWMA and its result placed on the Remote Egress Status Bus.

13.14.2.6 Egress P1 Twin Count EWMA Threshold Register (E_P1_Twin_EWMA_Th)

This register contains the threshold value that is compared against the Egress P1 Twin Count EWMA . The results of this comparison are placed on the Remote Egress Status Bus.

Access Type Read/Write

Base Addresses x‘A040 0800’



Field Name	Bit(s)	Reset	Description
Reserved	31:19		Reserved
P1_Twin_EWMA_Th	18:0	x‘0 0000’	Egress Priority 1 Twin Count Exponentially Weighted Moving Average Threshold value. This value is compared against the Egress P1 Twin Count EWMA and its result placed on the Remote Egress Status Bus.

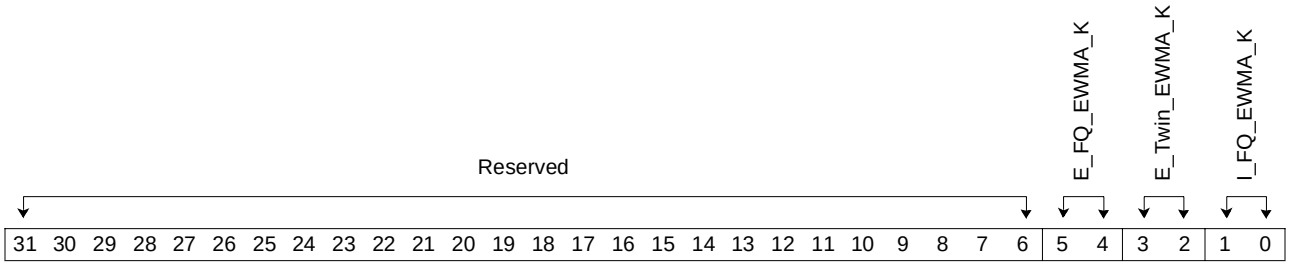
13.14.3 Exponentially Weighted Moving Average Constant (K) Register (EWMA_K)

This register contains Constant (K) values for the various Exponentially Weighted Moving Averages calculated in the Ingress and Egress Flow Control Hardware. The K value is encoded as follows:

K encoding	Constant Value
00	1
01	1/2
10	1/4
11	1/8

Access Type Read/Write

Base Addresses x‘A040 0040’



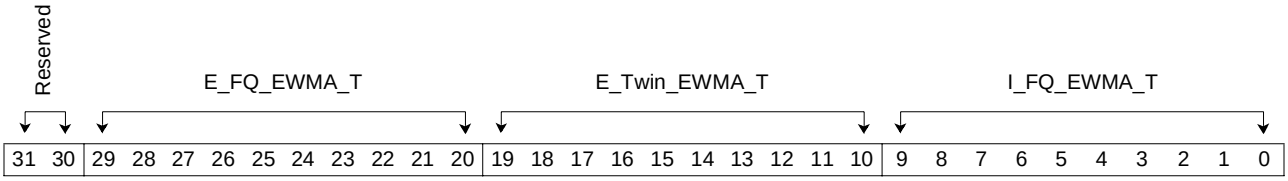
Field Name	Bit(s)	Reset	Description
Reserved	31:6		Reserved
E_FQ_EWMA_K	5:4	00	K value for the Egress Free Queue Count Exponentially Weighted Moving Average calculation in the Egress Flow Control Hardware.
E_Twin_EWMA_K	3:2	00	K value for the Egress P0/P1 Twin Count EWMA calculation in the Egress Flow Control Hardware.
I_FQ_EWMA_K	1:0	00	K value for the Ingress Free Queue Count Exponentially Weighted Moving Average calculation in the Ingress Flow Control Hardware.

13.14.4 Exponentially Weighted Moving Average Sample Period (T) Register (EWMA_T)

This register contains the sample periods for the various Exponentially Weighted Moving Averages calculated in the Ingress and Egress Flow Control Hardware. The values in this register are the number of 10 µs multiples for the interval between calculations of the respective ExpWAs. The computation of an EWMA does not occur unless the respective field in this register is non-zero.

Access Type Read/Write

Base Addresses x‘A040 0080’



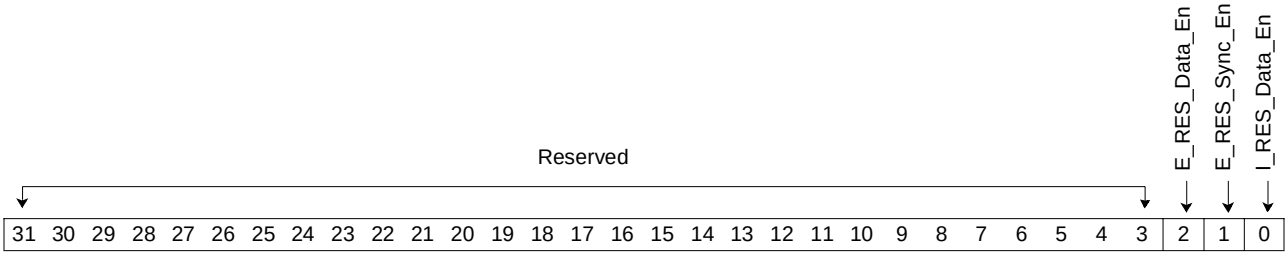
Field Name	Bit(s)	Reset	Description
Reserved	31:30		Reserved
E_FQ_EWMA_T	29:20	x‘000’	Sample Period for the Egress Free Queue Count Exponentially Weighted Moving Average calculation in the Egress Flow Control Hardware.
E_Twin_EWMA_T	19:10	x‘000’	Sample Period for the Egress P0/P1 Twin Count EWMA calculation in the Egress Flow Control Hardware.
I_FQ_EWMA_T	9:0	x‘000’	Sample Period for the Ingress Free Queue Count Exponentially Weighted Moving Average calculation in the Ingress Flow Control Hardware.

13.14.5 Remote Egress Status Bus Configuration Enables (RES_Data_Cnf)

This register controls operation of the Remote Egress Status Bus. The Remote Egress Status Bus is a 2-bit bus that allows communication between the system made up of all of the Egress Flow Control Hardware components and the system made up of all of the Ingress Flow Control Hardware. One bit of this bus is the sync pulse, and the other bit is TDM Data reflecting the status of the Egress Leased Twin Counts as they relate to their respective thresholds.

Access Type Read/Write

Base Addresses x'A000 0880'



Field Name	Bit(s)	Reset	Description
Reserved	31:3		Reserved
E_RES_Data_En	2	1	Egress Remote Egress Status Bus Data enable. 0 Sets the RES_Data IO to a Hi-Z state. 1 Places data about this NP's egress data store congestion state.
E_RES_Sync_En	1	0	Egress Remote Egress Status Bus Sync enable. When this field is set to 1, the NP4GS3 transmits a sync pulse every Remote Egress Status Bus interval. Only one network processor in a system will have this bit enabled.
I_RES_Data_En	0	1	Ingress Remote Egress Status Bus Data enable. 0 Disables use of the Remote Egress Status Bus for ingress flow control. The Ingress Flow Control Hardware treats the Remote Egress Status Bus as if it contained all 0s. 1 Enables use of the Remote Egress Status Bus for ingress flow control. Values are captured for each remote target blade and used for ingress flow control.

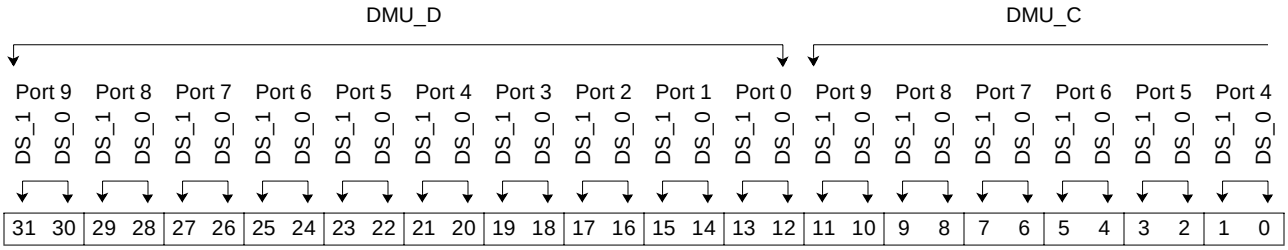
13.15 Target Port Data Storage Map (TP_DS_MAP) Register

The Target Port Data Storage Map indicates in which data store, DS_0 or DS_1 or both, that data is found for a port. Each port is configured with 2 bits; when set to 1, it indicates the data is found in the corresponding data store.

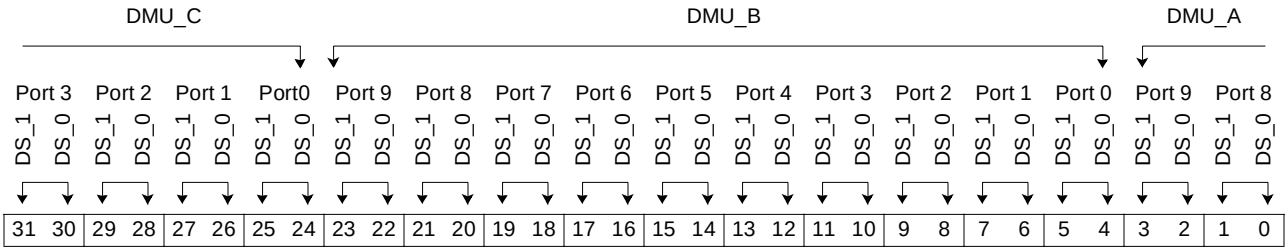
Access Type Read/Write

Base Address x‘A000 0140’

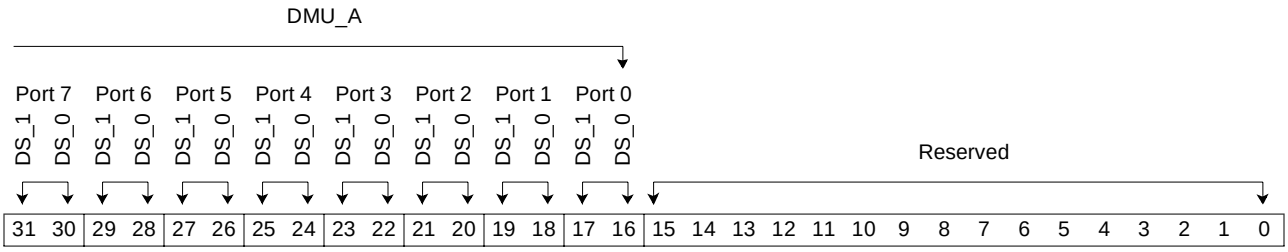
Base Address Offset 0



Base Address Offset 1



Base Address Offset 2



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
DMU_D Port 9	31:30	01	The relationship between individual bits and the datastore is shown in detail in the diagram above.
DMU_D Port 8	29:28	01	The relationship between individual bits and the datastore is shown in detail in the diagram above.
DMU_D Port 7	27:26	01	
DMU_D Port 6	25:24	01	
DMU_D Port 5	23:22	01	
DMU_D Port 4	21:20	01	
DMU_D Port 3	19:18	01	
DMU_D Port 2	17:16	01	
DMU_D Port 1	15:14	01	
DMU_D Port 0	13:12	01	
DMU_C Port 9	11:10	01	
DMU_C Port 8	9:8	01	
DMU_C Port 7	7:6	01	
DMU_C Port 6	5:4	01	
DMU_C Port 5	3:2	01	
DMU_C Port 4	1:0	01	

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
DMU_C Port 3	31:30	01	The relationship between individual bits and the datastore is shown in detail in the diagram above.
DMU_C Port 2	29:28	01	
DMU_C Port 1	27:26	01	
DMU_C Port 0	25:24	01	
DMU_B Port 9	23:22	01	
DMU_B Port 8	21:20	01	
DMU_B Port 7	19:18	01	
DMU_B Port 6	17:16	01	
DMU_B Port 5	15:14	01	
DMU_B Port 4	13:12	01	
DMU_B Port 3	11:10	01	
DMU_B Port 2	9:8	01	
DMU_B Port 1	7:6	01	
DMU_B Port 0	5:4	01	
DMU_A Port 9	3:2	01	
DMU_A Port 8	1:0	01	

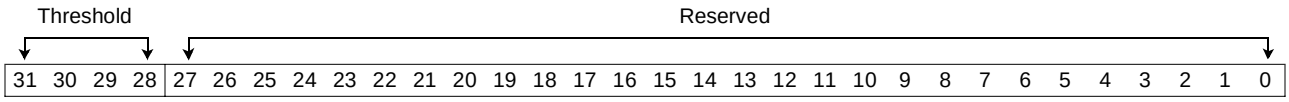
Base Address Offset 2

Field Name	Bit(s)	Reset	Description
DMU_A Port 7	31:30	01	The relationship between individual bits and the datastore is shown in detail in the diagram above.
DMU_A Port 6	29:28	01	
DMU_A Port 5	27:26	01	
DMU_A Port 4	25:24	01	
DMU_A Port 3	23:22	01	
DMU_A Port 2	21:20	01	
DMU_A Port 1	19:18	01	
DMU_A Port 0	17:16	01	
Reserved	15:0		Reserved

13.16 Egress SDM Stack Threshold Register (E_SDM_Stack_Th)

Access Type Read/Write

Base Address x‘A000 1800’



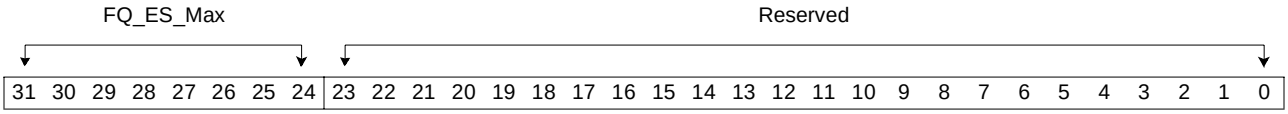
Field Name	Bit(s)	Reset	Description
Threshold	31:28	x‘8’	E-SDM Stack Threshold value. When this threshold is violated (threshold value is less than the count of empty entries in the E-SDM Stack), send grant is set to its disable state.
Reserved	27:0		Reserved

13.17 Free Queue Extended Stack Maximum Size (FQ_ES_Max) Register

This register sets the number of buffers that are released into the free queue and thus made available for the storage of received frames. The egress EDS reads this register when building the FQ_ES.

Access Type Read/Write

Base Address x‘A000 2100’



Field Name	Bit(s)	Reset	Description
FQ_ES_Max	31:24	x‘08’	Maximum size of the Free Queue Extended Stack measured in increments of 2 K buffer twins. The Egress EDS reads this value when building the FQ_ES. The maximum size is limited by the DDR DRAM used by the egress data store. Each 128-bit page holds six entries each. Once this register is written, the hardware creates entries in the Buffer Free queue (FQ) at a rate of 6 entries every 150 or 165 ns (rate is dependent on the setting of bit 6 of the DRAM Parameter register - 11/$\overline{10}$). The value in this register may be modified during operation. However, the new value may not be smaller than the current value.
Reserved	23:0		Reserved

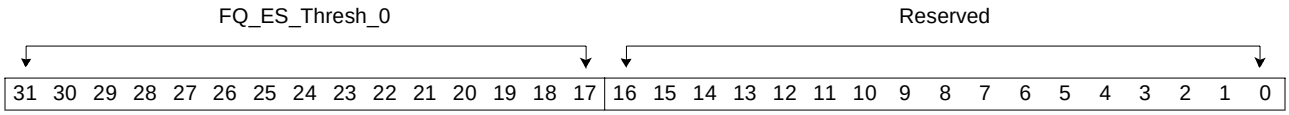
13.18 Egress Free Queue Thresholds

A queue count is maintained by the free queue extended stack management hardware. The count value is continuously compared to the value contained in each of three threshold registers. The result of this comparison affects the NP4GS3's flow control mechanisms. The register values must be chosen such that FQ_ES_Th_0 ≤ FQ_ES_Th_1 ≤ FQ_ES_Th_2.

13.18.1 FQ_ES_Threshold_0 Register (FQ_ES_Th_0)

Access Type Read/Write

Base Address x'A000 2010'

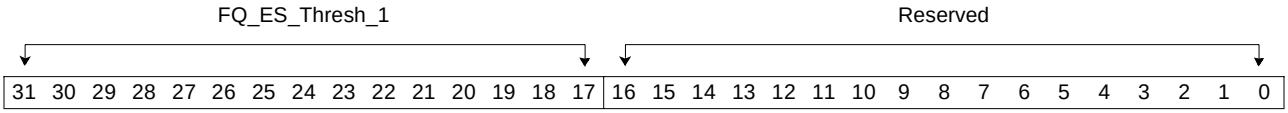


Field Name	Bit(s)	Reset	Description
FQ_ES_Thresh_0	31:17	x'0000'	Free Queue Extended Stack Threshold 0 as measured in units of 16 Twins. When this threshold is violated (the threshold value is greater than the number of remaining twins in the free queue): • Frame data received at the switch interface is discarded (the number of frames discarded is counted). • Frames that have started re-assembly that receive data while this threshold is violated are also discarded (all data associated with the frame is discarded). • Guided traffic data is not discarded. • An interrupt is sent to the EPC.
Reserved	16:0		Reserved

13.18.2 FQ_ES_Threshold_1 Register (FQ_ES_Th_1)

Access Type Read/Write

Base Address x‘A000 2020’

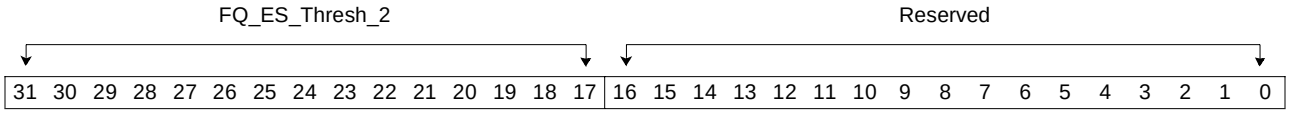


Field Name	Bit(s)	Reset	Description
FQ_ES_Thresh_1	31:17	x‘0000’	Free Queue Extended Stack Threshold 1 as measured in units of 16 Twins. When this threshold is violated (the threshold value is greater than the number of remaining twins in the free queue), an interrupt is sent to the EPC.
Reserved	16:0		Reserved

13.18.3 FQ_ES_Threshold_2 Register (FQ_ES_Th_2)

Access Type Read/Write

Base Address x‘A000 2040’



Field Name	Bit(s)	Reset	Description
FQ_ES_Thresh_2	31:17	x‘0000’	Free Queue Extended Stack Threshold 2 as measured in units of 16 Twins. When this threshold is violated (the threshold value is greater than the number of remaining twins in the free queue), an interrupt is sent to the EPC and, if enabled by DMU configuration, the Ethernet preamble is reduced to 6 bytes
Reserved	16:0		Reserved

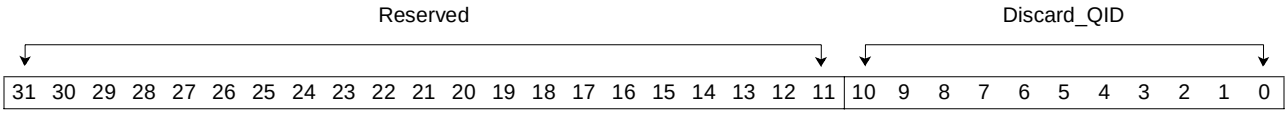
13.19 Discard Flow QCB Register (Discard_QCB)

This register is used by the egress hardware when the scheduler and flow control are enabled. This register contains the address of the flow QCB to be used when egress flow control actions require that the frame be discarded. This register and the QCB referenced by the address must be configured by hardware. See *Table 53* on page 147 for details on configuring the QCB.

When the scheduler is disabled, the register contains the target port queue to which the flow control discarded frames are sent. This value should be set to x‘029’.

Access Type Read/Write

Base Address x‘A000 1400’



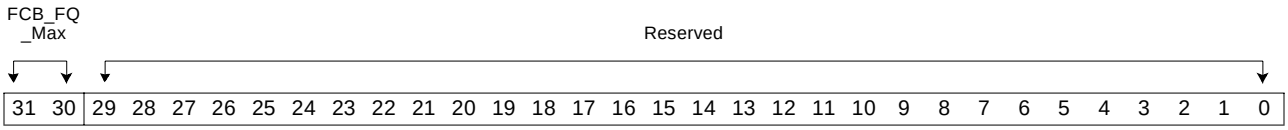
Field Name	Bit(s)	Reset	Description
Reserved	31:11		Reserved
Discard_QID	10:0	x‘029’	The Discard QID field contains the address of the QCB that has been configured for discarding egress frames while the scheduler is enabled. When the scheduler is disabled, this is the target port ID (x‘029’) to which discarded frames due to flow control discard actions are sent.

13.20 Frame Control Block FQ Size Register (FCB_FQ_Max)

This register sets the number of Frame control blocks that are released to the FCB FQ during initialization.

Access Type Read/Write

Base Address x‘A000 2200’



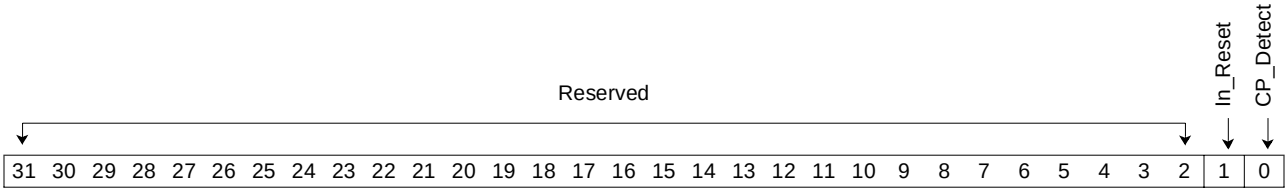
Field Name	Bit(s)	Reset	Description
FCB_FQ_Max	31:30	00	Indicates the number of FCBs released into the FCB free queue during initialization. 00 128K 01 256K 10 512K 11 1M
Reserved	29:0		Reserved

13.21 Data Mover Unit (DMU) Configuration

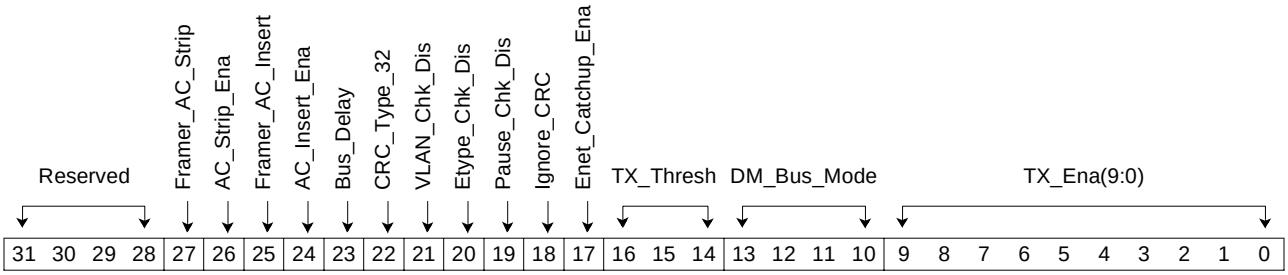
There are four Data Mover Units (DMU) configured for internal MAC operation, for external connection detection (i.e. attached control point detection), and for external bus operation (i.e TMII, SMII, TBI, and POS framer).

Access Type	Base Address Offset 0	Read only
	Base Address Offset 1	Read/Write
	Base Address Offset 2	Read/Write
	Base Address Offset 3	Read/Write
Base Address	DMU_A	x‘A001 0010’
	DMU_B	x‘A001 0020’
	DMU_C	x‘A001 0040’
	DMU_D	x‘A001 0080’

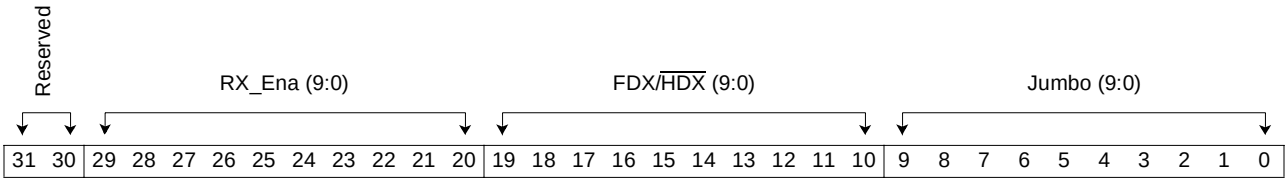
Base Address Offset 0



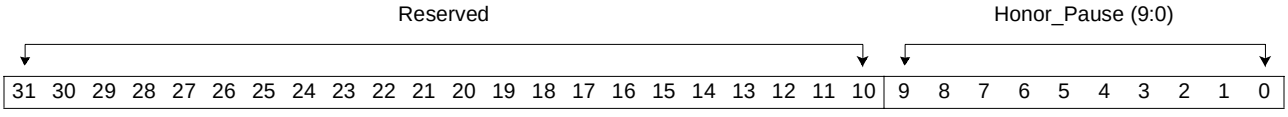
Base Address Offset 1



Base Address Offset 2



Base Address Offset 3



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
Reserved	31:2		Reserved
In_Reset	1	1	DMU In Reset indicator originates in the clocking logic. 0 Written when the clock logic removes the reset signal for the DMU. 1 DMU is held in reset mode.
CP_Detect	0	0	Control Point Detected indicator value originates in the PMM. 0 Control point processor connection not present 1 DMU detected a Control Point processor connection.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
Reserved	31:28		Reserved
Framer_AC_Strip	27	0	Configures the MAC operation for an attached POS framer. 0 Framer passes AC field to the network processor 1 Framer does not pass AC field to the network processor. The CRC checking performed is modified to account for the missing field. AC value of x'FF03' is assumed.
AC_Strip_Ena	26	0	Configures the MAC operation for an attached POS framer. 0 AC field is not stripped from the packet. 1 AC field is stripped from the packet prior to being stored in the ingress data store.
Framer_AC_Insert	25	0	Configures the MAC operation for an attached POS framer. 0 AC field is assumed present in the packet sent to the framer. 1 AC field is not present in the packet sent to the framer. The framer inserts this field and CRC generation is adjusted to account for the missing AC field. An AC value of x'FF03' is assumed.
AC_Insert_Ena	24	1	Configures the MAC operation for an attached POS framer. 0 AC field is not inserted by the MAC. For proper operation, Framer_AC_Insert must be set to 1. 1 AC field is inserted by the MAC. For proper operation, Framer_AC_Insert must be set to 0.
Bus_Delay	23	0	Bus Delay controls the length of the delay between a poll request being made and when the MAC samples the framer's response. 0 Sample is taken 1 cycle after the poll request 1 Sample is taken 2 cycles after the poll request.
CRC_Type_32	22	0	CRC_Type_32 controls the type of frame CRC checking and generation performed in the MAC. 0 16-bit CRC in use. 1 32-bit CRC in use
VLAN_Chk_Dis	21	0	VLAN Checking Disable control value. 0 Enable VLAN checking. 1 Disable VLAN checking by the DMU.
Etype_Chk_Dis	20	0	Ethernet Type Checking Disable control value. 0 Enable DMU checking. 1 Disable DMU checking of E_Type_C and E_Type_D

Field Name	Bit(s)	Reset	Description
Pause_Chk_Dis	19	0	Pause Frame Checking Disable control value. 0 Pause frames are processed by the MAC. They are not sent to the Ingress EDS. 1 Pause frames are not processed by the MAC. The frames are sent to the Ingress EDS for service. See also Honor_Pause in offset 3 for additional control of the MAC in relation to pause frames.
Ignore_CRC	18	0	Ignore CRC controls the behavior of CRC checking for each DMU. 0 Discard frames with bad CRC 1 Ignore bad CRC
Enet_Catchup_Ena	17	1	Ethernet MAC catch up enabled. When enabled and FQ_ES_Threshold_2 is violated, the MAC uses a 6-byte preamble instead of a 7-byte preamble. 0 Disabled 1 Enabled
TX_Thresh	16:14	100	Transmit Threshold configures the number of cell buffers that must be filled before the transmission of a frame can start. 000 Invalid 001-100 Valid range 101-111 Invalid
DM_Bus_Mode	13:10	1010	Data Mover Bus Mode configures the mode in which the DM Bus operates. 0000 Reserved 0001 10/100 Ethernet SMII Mode 0010 Gigabit Ethernet GMII Mode 0011 Gigabit Ethernet TBI Mode 0100 POS OC12 Mode; non-polling POS support 0101 POS 4xOC3 mode; polling POS support 0110-1001 Reserved 1010 CP Detect Mode 1011 Debug Mode (DMU D only) 1100-1101 Reserved 1110 DMU Disabled 1111 POS OC48 Mode; Quad DMU mode, non-polling, POS support
TX_Ena(9:0)	9:0	0	Port Transmit Enable control is a bitwise enable for each port's transmitter. 0 Disable Port 1 Enable Port

Base Address Offset 2

Field Name	Bit(s)	Reset	Description
Reserved	31:30		Reserved
RX_Ena(9:0)	29:20	0	Port Receive Enable control is a bitwise enable for each port's receiver. 0 Disable Port 1 Enable Port
FDX/$\overline{\text{HDX}}$(9:0)	19:10	0	Full Duplex or Half Duplex operation mode for ports 9 to 0 controls the mode of operation for the associated port. 0 Half Duplex (HDX) operation 1 Full Duplex (FDX) operation

Field Name	Bit(s)	Reset	Description
Jumbo(9:0)	9:0	0	Jumbo frame operation mode for ports 9 to 0 controls the mode of operation for the associated port. 0 Jumbo Frames Disabled 1 Jumbo Frames Enabled

Base Address Offset 3

Field Name	Bit(s)	Reset	Description
Reserved	31:10		Reserved
Honor_Pause(9:0)	9:0	0	Honor Pause control value is a bitwise control value for the port's pause function. 0 Ignore pause frames received by corresponding port 1 Pause when pause frame received by corresponding port

13.22 QD Accuracy Register (QD_Acc)

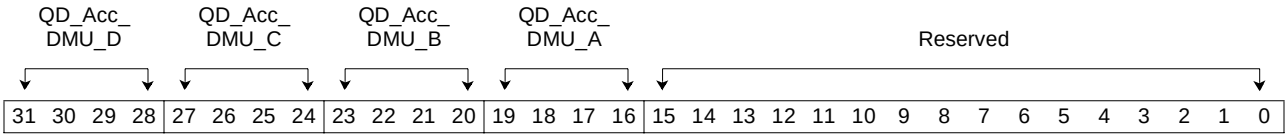
The QD Accuracy register tunes the egress scheduler's WFQ rings. The values assure fairness and some benefit to queues with lower defined QD values which expect better service when enqueueing to an empty queue. The value is also a scaling factor when servicing queues. Configuration recommendations are dependent on the maximum frame sizes expected for a DMU.

Max Frame size	QD_Acc_DMU
2 K	6
9 K	8
14 K	10

There is one field defined per media DMU (A-D).

Access Type Read/Write

Base Address x'A002 4000'



Field Name	Bit(s)	Reset	Description
QD_Acc_DMU_D	31:28	0	QD Accuracy value used for DMU_D
QD_Acc_DMU_C	27:24	0	QD Accuracy value used for DMU_C
QD_Acc_DMU_B	23:20	0	QD Accuracy value used for DMU_B
QD_Acc_DMU_A	19:16	0	QD Accuracy value used for DMU_A
Reserved	15:0		Reserved

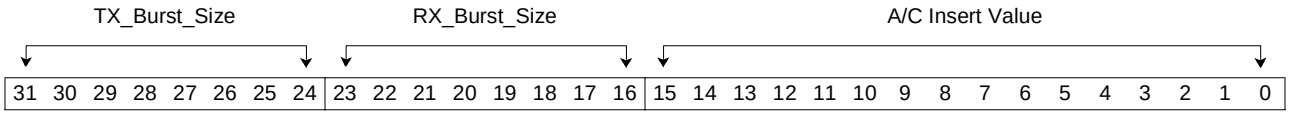
13.23 Packet Over SONET Control Register (POS_Ctrl)

One configuration register per DMU is provided to control POS framer interaction. It configures transmit and receive burst sizes and sets the value used for AC field insertion.

Access Type Read/Write

Base Address

DMU_A	x‘A004 0100’
DMU_B	x‘A004 0200’
DMU_C	x‘A004 0400’
DMU_D	x‘A004 0800’



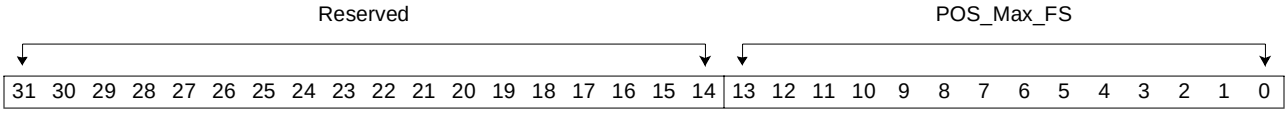
Field Name	Bit(s)	Reset	Description
TX_Burst_Size	31:24	x'10'	Transmit burst size. Used only for OC3 modes of operation. OC12 and OC48 interfaces transmit until the framer de-assertsTxPFA. When set to 0 the MAC uses TxPFA from the frame to stop transmission of data. When set to a value other than 0, the MAC will burst data to the framer up to the burst size or until the framer drops TxPFA. It is recommended that the low water mark in the frame be set to a value equal to or greater than the value of Tx_Burst_Size.
RX_Burst_Size	23:16	x'10'	Receive burst size
A/C Insert Value	15:0	x'FF03'	Value used by the MAC when AC_Insert_Ena is set to 1

13.24 Packet Over SONET Maximum Frame Size (POS_Max_FS)

This register controls the maximum frame size supported by the network processor. POS permits frames up to 64 K bytes, however, the network processor is constrained to 14 K (14336) bytes maximum. This register allows setting for smaller frame sizes. Frames received by the network processor that exceed the length specified by this register are aborted during reception.

Access Type Read/Write

Base Address x‘A004 0080’



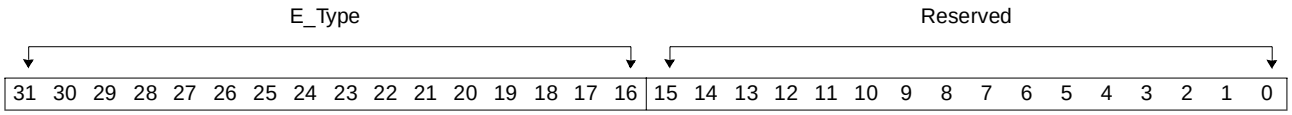
Field Name	Bit(s)	Reset	Description
Reserved	31:14		Reserved
POS_Max_FS	13:0	x‘3800’	Packet over SONET Maximum Frame Size sets the maximum frame size that the network processor can receive on a POS port. The value in this register is used to determine the length of a long frame for the ingress and egress Long Frame counters.

13.25 Ethernet Encapsulation Type Register for Control (E_Type_C)

This configuration register is used by the PMM to recognize Ethernet-encapsulated guided frames. When the Ethernet frame's type field matches this value, the MAC DA, SA, and Type fields of the frame are stripped and the frame is queued onto the GFQ.

Access Type Read/Write

Base Addresses x'A001 1000'



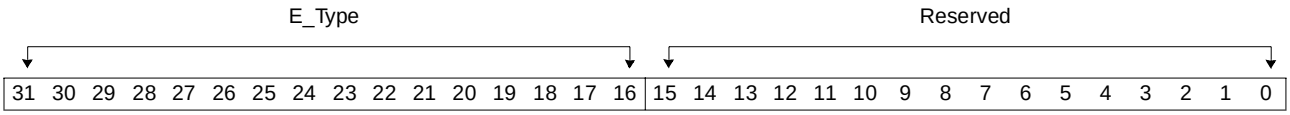
Field Name	Bit(s)	Reset	Description
E_Type	31:16	x'0000'	Ethernet Type used for encapsulated guided traffic.
Reserved	15:0		Reserved

13.26 Ethernet Encapsulation Type Register for Data (E_Type_D)

This configuration register is used by the PMM to recognize Ethernet-encapsulated data frames. When the Ethernet frame's type field matches this value, the MAC DA, SA, and Type fields of the frame are stripped and the frame is queued onto the GDQ.

Access Type Read/Write

Base Addresses x'A001 2000'



Field Name	Bit(s)	Reset	Description
E_Type	31:16	x'0000'	Ethernet Type used for encapsulated CP data traffic.
Reserved	15:0		Reserved

13.27 Source Address Array (SA_Array)

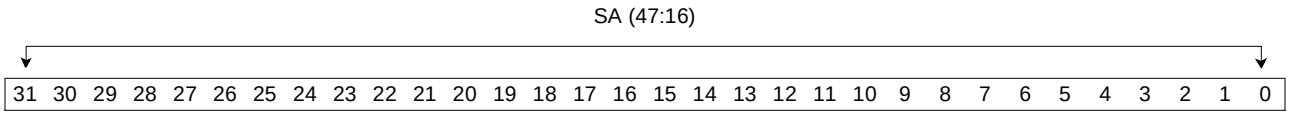
The SA Array is a register array containing 64 Source Address values. The SA Pointer from the egress FCB references elements of this register array. The value retrieved from the SA Array is used to insert or overlay the SA field of transmitted frames during egress frame alteration.

Access Type Read/Write

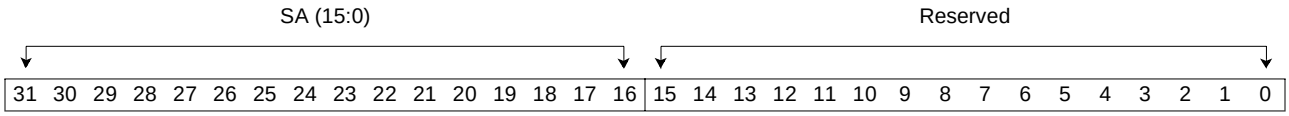
Base Addresses Note: Each DMU listed below contains 64 entries. Nibbles 5 and 6 of the base address is incremented by x‘01’ for each successive entry, represented by ‘##’, and ranging from x‘00’ to x‘3F’. The word offset is represented by ‘W’, and ranges from x‘0’ to x‘1’.

DMU_A	x‘8810 0##W’
DMU_B	x‘8811 0##W’
DMU_C	x‘8812 0##W’
DMU_D	x‘8813 0##W’
Wrap	x‘8814 0##W’

Base Address Offset 0



Base Address Offset 1



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
SA(47:16)	31:0	Not defined	Source Address value. The DMU accesses an SA value when it performs egress frame alteration functions. The data is the source address that is either overlaid or inserted into a frame. The entry address is inferred from the contents of the FCB.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
SA(15:0)	31:16	Not defined	Source Address value. The DMU accesses an SA value when it performs egress frame alteration functions. The data is the source address that is either overlaid or inserted into a frame. The entry address is inferred from the contents of the FCB.
Reserved	15:0		Reserved

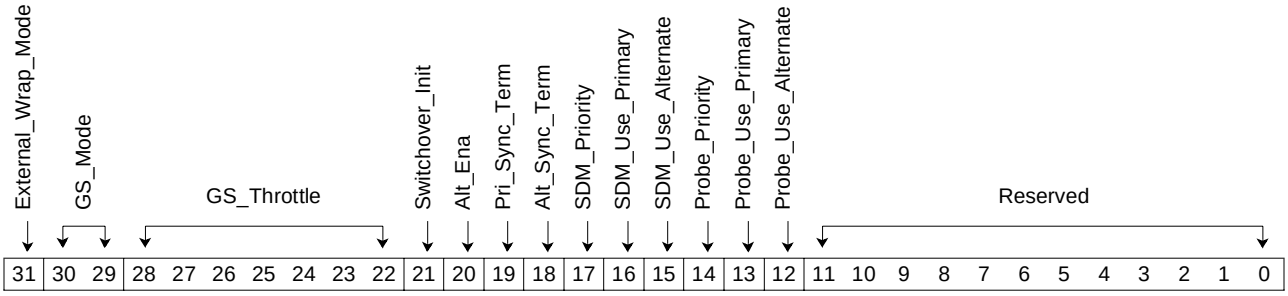
13.28 DASL Initialization and Configuration

13.28.1 DASL Configuration Register (DASL_Config)

This register contains control information for the DASL-A and DASL-B interfaces.

Access Type Read/Write

Base Address x'A000 0110'



Field Name	Bit(s)	Reset	Description
External_Wrap_Mode	31	0	External Wrap Mode indicator value. 0 The network processor is configured for DASL connection to one or two switches supporting up to 64 target blades. 1 The network processor is configured for external DASL wrap connections. One or two network processors can be supported in this mode, restricting target blade addressing to the values of 0 or 1.
GS_Mode	30:29	01	Grant Status Mode controls the setting of and response to grant status by a functional unit. 00 Switch Grant Status Algorithm 01 Deassert Grant Status when E-SDM is Full. Ingress EDS stops sending cells to the SWI. 10 Deassert Grant Status one out of every N+1 times, where the value N is contained in the GS Throttle field. 11 Deassert Grant Status when E-SDM if Full. I-SIF I-SCI stops sending cells to the DASL.
GS_Throttle	28:22	x'00'	Grant Status Throttle When GS Mode is set to '10', GS throttle controls the rate at which Grant Status is deasserted. That rate is once every "GS_Throttle + 1" cell times.
Switchover_Init	21	0	Switchover Initialization 0 Nop 1 Switchover Initialization re-starts the Primary DASL interface (normal use is in response to a switchover event).
Alt_Ena	20	0	Alternate DASL Enable control flag. 0 Nop 1 Set by SWI to enable the Alternate DASL Port. Then the SWI starts sending Sync Cells and the receiver searches the attain synchronization with the incoming serial stream.
Pri_Sync_Term	19	0	Primary DASL Synchronization Termination 0 Nop 1 Stops the NP4GS3's primary DASL interface from sending the synchronization pattern that completes the DASL synchronization sequence on the primary DASL interface.

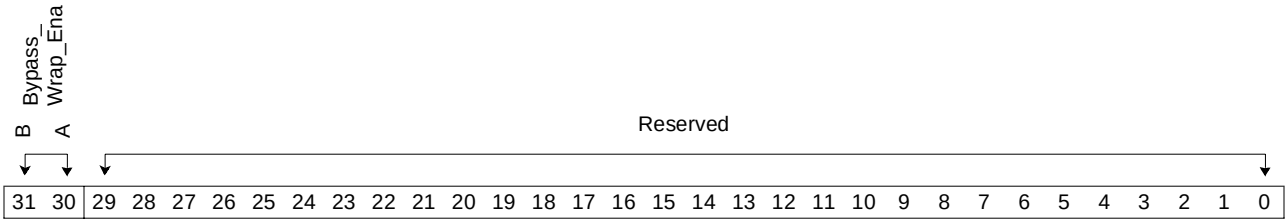
Field Name	Bit(s)	Reset	Description
Alt_Sync_Term	18	0	Alternate DASL Synchronization Termination 0 Nop 1 Stops the NP4GS3's alternate DASL interface from sending the synchronization pattern that completes the DASL synchronization sequence on the alternate DASL interface.
SDM_Priority	17	1	I-SDM Priority configures the arbitration priority of I-SDM accesses to the DASL interface. 0 High priority 1 Low priority
SDM Use Primary	16	1	SDM Use Primary 0 Nop 1 Configures the I-SDM traffic to use the primary DASL interface.
SDM_Use_Alternate	15	0	SDM Use Alternate 0 Nop 1 Configures the I-SDM traffic to use the alternate DASL interface.
Probe_Priority	14	0	Probe_Priority configures the arbitration priority of the Probe accesses to the DASL interface. 0 High priority 1 Low priority
Probe_Use_Primary	13	0	Probe Use Primary 0 Nop 1 Configures the Probe traffic to use the primary DASL interface.
Probe_Use_Alternate	12	1	Probe Use Alternate 0 Nop 1 Configures the Probe traffic to use the alternate DASL interface.
Reserved	11:0		Reserved

13.28.2 DASL Bypass and Wrap Register (DASL_Bypass_Wrap)

This register controls the internal wrap path which bypasses the DASL interface.

Access Type Read/Write

Base Address x‘A002 0080’



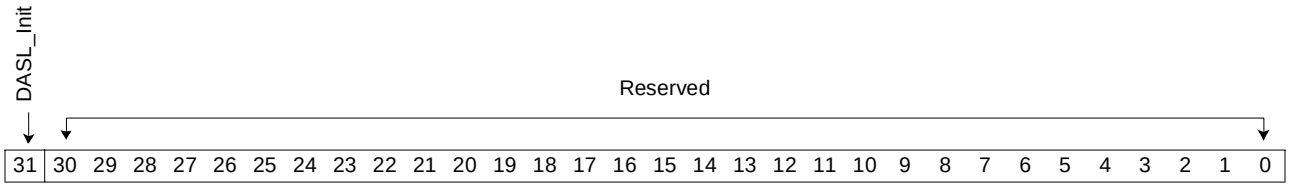
Field Name	Bit(s)	Reset	Description
Bypass_Wrap_Ena	31:30	00	DASL bypass and wrap control enables or disables the DASL bypass and internal wrap path. Bit 31 controls the wrap for the DASL-B interface. Bit 30 controls the wrap for the DASL-A interface. 0 Wrap disabled 1 Wrap enabled
Reserved	29:0		Reserved

13.28.3 DASL Start Register (DASL_Start)

This configuration register initializes the DASL interfaces when the DASL_Init field makes a transition from '0' to '1'. The completion of DASL initialization is reported in the Init_Done Register.

Access Type Read/Write

Base Address x'A000 0210'



Field Name	Bit(s)	Reset	Description
DASL_Init	31	0	DASL initialization control value. Switching this value from '0' to '1' causes the DASL interfaces to initialize.
Reserved	30:0		Reserved

14. Electrical and Thermal Specifications

The NP4GS3 utilizes IBM CMOS SA27E technology.

Table 223: Absolute Maximum Ratings

Symbol	Parameter	Rating	Units	Notes
		1.8 V		
V_{DD}	Power Supply Voltage	1.95	V	1
T_A	Operating Temperature (ambient)	-40 to +100	°C	1
T_J	Junction Temperature	-55 to +125	°C	1
T_{STG}	Storage Temperature	-65 to +150	°C	1

1. Stresses greater than those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

Table 224: Input Capacitance (pF) (Page 1 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
A02	SCH_Addr(11)	11.31
A03	Spare_Tst_Rcvr(2)	9.37
A04	SCH_Addr(02)	11.01
A05	$\overline{\text{Switch\_Clk\_B}}$	8.67
A06	SCH_Data(08)	10.81
A07	SCH_Addr(00)	10.21
A08	SCH_Data(15)	10.31
A09	SCH_Data(07)	10.01
A10	SCH_Data(00)	9.91
A11	D4_Addr(01)	9.81
A12	DD_BA(0)	10.01
A13	D4_Data(31)	8.8
A14	D4_Data(24)	8.5
A15	D4_Data(19)	8.7
A16	D4_Data(11)	8.5
A17	D4_Data(04)	8.7
A18	DS1_Data(26)	8.5
A19	DS1_Data(19)	8.8
A20	DS1_Data(14)	8.7
A21	DS1_Data(06)	9
A22	DS1_Data(00)	9.1
A23	DS0_Addr(10)	10.21

Table 224: Input Capacitance (pF) (Continued) (Page 2 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
A24	DS0_DQS(2)	9
A25	DS0_Data(28)	9.2
A26	DS0_Data(20)	9.5
A27	DS0_Data(14)	9.5
A28	DS0_Addr(00)	11.41
A29	DS0_Data(09)	10
A30	DS0_Data(13)	10.4
A31	DS0_CS	11.81
A32	DS0_Data(01)	10.8
A33	DS0_Data(03)	11.7
AA03	DASL_In_A(7)	6.5
AA04	$\overline{\text{DASL\_In\_A(6)}}$	5.9
AA05	LU_Addr(17)	7.31
AA06	LU_Data(23)	7.71
AA07	LU_Data(28)	7.51
AA08	LU_Data(29)	7.01
AA09	LU_Addr(00)	6.81
AA10	LU_Addr(02)	6.31
AA11	LU_Addr(18)	5.81
AA12	D0_Data(00)	4.8
AA14	D0_Addr(04)	6.11
AA15	D1_Data(06)	5
AA16	D3_Data(01)	5.1
AA17	D3_Addr(06)	6.21
AA18	D2_Data(03)	5
AA19	D2_Addr(12)	6.21
AA20	DA_BA(1)	6.1
AA22	JTAG_TCk	5.85
AA23	JTAG_TDO	6.15
AA25	DMU_A(28)	7.05
AA26	DMU_A(27)	7.25
AA27	DMU_A(26)	7.15
AA28	DMU_A(25)	7.55
AA29	DMU_A(24)	7.85
AA30	DMU_A(23)	8.45
AA31	DMU_A(22)	8.95

Table 224: Input Capacitance (pF) (Continued) (Page 3 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AA32	DMU_A(21)	9.55
AA33	DMU_A(20)	9.95
AB03	DASL_In_A(6)	6.4
AB05	LU_Addr(14)	7.81
AB07	LU_Addr(03)	7.51
AB09	LU_Addr(16)	6.91
AB11	D0_Data(03)	4.7
AB13	D0_Data(28)	5
AB15	D0_Addr(05)	6.11
AB17	D1_Addr(00)	6.21
AB19	D2_DQS(0)	5.1
AB21	D6_Data(04)	5
AB23	$\overline{\text{C405\_Debug\_Halt}}$	6.15
AB25	PCI_AD(02)	8.55
AB27	PCI_AD(01)	8.55
AB29	PCI_AD(00)	9.75
AB31	DMU_A(30)	8.85
AB33	DMU_A(29)	9.95
AC01	LU_Addr(07)	9.41
AC02	DASL_In_A(5)	7
AC03	$\overline{\text{DASL\_In\_A(5)}}$	6.6
AC04	LU_Addr(11)	8.41
AC08	LU_R_$\overline{\text{Wrt}}$	7.51
AC09	LU_Addr(04)	7.01
AC11	D0_Data(13)	5
AC12	D0_DQS(1)	5
AC14	D0_Addr(10)	6.31
AC15	D0_Data(29)	5.1
AC16	D1_Data(15)	5.1
AC17	D3_DQS(1)	5.1
AC18	D3_Addr(07)	6.41
AC20	D2_Addr(05)	6.51
AC21	D6_Data(10)	5.1
AC22	D6_Data(15)	5.1
AC23	PCI_Bus_M_Int	7.45
AC24	PCI_AD(11)	8.15

Table 224: Input Capacitance (pF) (Continued) (Page 4 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AC25	PCI_AD(10)	8.55
AC26	PCI_AD(09)	8.45
AC27	PCI_AD(08)	8.65
AC28	PCI_CBE(0)	9.25
AC29	PCI_AD(07)	9.65
AC30	PCI_AD(06)	10.35
AC31	PCI_AD(05)	10.45
AC32	PCI_AD(04)	10.85
AC33	PCI_AD(03)	11.35
AD01	$\overline{\text{DASL\_In\_A(4)}}$	7.6
AD03	DASL_In_A(4)	6.8
AD07	LU_Addr(12)	7.71
AD09	D0_Data(01)	5.9
AD11	D0_Data(23)	5.2
AD13	DB_BA(1)	6.5
AD15	D1_CS	6.61
AD19	D3_WE	6.81
AD21	D2_Addr(06)	6.81
AD25	PCI_CBE(1)	8.55
AD27	PCI_AD(15)	9.45
AD29	PCI_AD(14)	9.95
AD31	PCI_AD(13)	10.65
AD33	PCI_AD(12)	11.45
AE01	DASL_In_A(3)	7.7
AE02	$\overline{\text{DASL\_In\_A(1)}}$	7.2
AE03	$\overline{\text{DASL\_In\_A(3)}}$	6.9
AE06	LU_Addr(05)	8.61
AE07	LU_Addr(06)	7.61
AE08	LU_Addr(15)	7.71
AE09	D0_Data(11)	5.9
AE10	D0_Data(22)	5.9
AE11	D0_DQS(2)	5.8
AE12	D1_Data(00)	5.7
AE13	DB_RAS	6.7
AE14	D1_Addr(07)	6.91
AE15	D3_Data(03)	5.8

Table 224: Input Capacitance (pF) (Continued) (Page 5 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AE16	D3_Data(09)	5.7
AE17	D2_Data(08)	6.3
AE18	D3_Addr(05)	7.11
AE19	D3_Addr(12)	7.21
AE20	D2_Data(07)	6
AE21	D2_Data(09)	6
AE22	D6_Data(14)	6
AE23	D6_Data(09)	6.1
AE24	D6_Addr(02)	7.3
AE25	MGrant_B(1)	7.15
AE26	PCI_Frame	8.75
AE27	PCI_IRdy	9.65
AE28	PCI_TRdy	10.25
AE29	PCI_DevSel	10.05
AE30	PCI_Stop	10.55
AE31	PCI_PErr	10.75
AE32	PCI_SErr	11.05
AE33	PCI_Par	11.55
AF01	DASL_In_A(1)	8
AF07	LU_Addr(13)	7.91
AF09	D0_Data(20)	6.5
AF11	D0_Addr(12)	7.51
AF13	D1_Addr(06)	7.21
AF15	D3_Data(14)	6.1
AF17	D3_Addr(02)	7.01
AF19	D3_Data(15)	6.2
AF21	D6_DQS_Par(01)	6.3
AF23	D6_ByteEn(1)	7.7
AF25	D6_Addr(04)	7.5
AF27	PCI_AD(17)	9.55
AF29	PCI_AD(16)	10.35
AF31	PCI_CBE(2)	11.15
AF33	PCI_Clk	11.85
AG03	D0_Data(09)	8
AG05	LU_Addr(09)	10.01
AG06	LU_Addr(10)	8.81

Table 224: Input Capacitance (pF) (Continued) (Page 6 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AG07	D0_Data(02)	6.7
AG08	D0_Data(21)	6.6
AG09	D0_DQS(0)	6.5
AG10	D0_Addr(11)	7.71
AG11	DB_BA(0)	6.4
AG12	D1_Addr(05)	7.61
AG13	D3_Data(00)	6.5
AG14	D3_Data(02)	6.5
AG15	D3_Data(13)	6.4
AG16	D3_Data(10)	6.4
AG17	D3_Data(12)	6.2
AG18	D3_Addr(04)	7.71
AG19	D3_Addr(00)	7.71
AG20	D3_DQS(0)	6
AG21	D6_Addr(11)	7.1
AG22	D2_Data(10)	6.1
AG23	D2_Addr(07)	7.61
AG24	D6_ByteEn(0)	8.2
AG25	DA_RAS	8.2
AG26	D6_Addr(03)	8.3
AG27	MGrant_B(0)	8.25
AG28	PCI_AD(23)	10.45
AG29	PCI_AD(22)	11.65
AG30	PCI_AD(21)	10.95
AG31	PCI_AD(20)	11.05
AG32	PCI_AD(19)	11.75
AG33	PCI_AD(18)	11.85
AH01	DASL_In_A(2)	8.7
AH03	$\overline{\text{DASL\_In\_A(2)}}$	7.9
AH05	DE_BA(1)	8.61
AH07	D0_Data(10)	7.7
AH09	D0_DQS(3)	6.7
AH11	D1_Data(11)	6.3
AH13	D1_Data(14)	6.2
AH15	D1_Addr(11)	7.41
AH17	D3_Data(11)	6.1

Table 224: Input Capacitance (pF) (Continued) (Page 7 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AH19	DB_Clk	7.4
AH21	D2_Addr(01)	7.71
AH23	D2_Addr(04)	8.01
AH25	D6_Data(08)	7.3
AH27	D6_Addr(12)	9.6
AH29	PCI_AD(24)	10.25
AH31	PCI_CBE(3)	11.65
AH33	PCI_IDSel	12.45
AJ02	$\overline{\text{DASL\_In\_A(0)}}$	8.5
AJ03	LU_Clk	9.51
AJ04	DE_Clk	9
AJ05	DE_$\overline{\text{Clk}}$	8.7
AJ06	D0_Data(14)	8.6
AJ07	D0_Data(16)	7.4
AJ08	D0_Data(31)	7.2
AJ09	D0_Addr(02)	8.31
AJ10	D1_Data(03)	6.9
AJ11	D1_Data(07)	6.4
AJ12	DB_CAS	7.8
AJ13	D1_Addr(01)	7.21
AJ14	D1_DQS(0)	6.7
AJ15	D1_Addr(12)	7.91
AJ16	D3_Addr(01)	7.61
AJ17	D3_Addr(03)	7.81
AJ18	DB_$\overline{\text{Clk}}$	7.4
AJ19	D2_Data(01)	6.8
AJ20	D2_Data(04)	6.9
AJ21	D2_Data(14)	7.1
AJ22	D2_Addr(00)	8.41
AJ23	D2_Addr(11)	8.61
AJ24	D2_WE	8.61
AJ25	D6_WE	8.7
AJ26	D6_Data(12)	7.9
AJ27	D6_Addr(07)	9.1
AJ28	D6_Addr(09)	10.4
AJ29	PCI_AD(29)	10.65

Table 224: Input Capacitance (pF) (Continued) (Page 8 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AJ30	PCI_AD(28)	10.95
AJ31	PCI_AD(27)	11.35
AJ32	PCI_AD(26)	12.15
AJ33	PCI_AD(25)	12.25
AK01	DASL_In_A(0)	9.3
AK03	DE_RAS	9.91
AK05	D0_Data(15)	8
AK07	D0_Data(30)	8.1
AK09	D1_Data(04)	7.7
AK11	D1_Data(08)	7.2
AK13	D1_Addr(02)	8.41
AK15	D3_Data(07)	7.3
AK17	D3_Addr(11)	8.11
AK19	D3_Addr(08)	8.71
AK21	D2_Data(13)	7.5
AK23	D2_Addr(10)	8.81
AK25	D2_DQS(1)	8.3
AK27	D6_Data(13)	8.7
AK29	D6_Addr(08)	9.8
AK31	PCI_AD(31)	11.65
AK33	PCI_AD(30)	12.95
AL01	Spare_Tst_Rcvr(4)	8.25
AL02	DE_CAS	10.61
AL03	D0_Data(12)	9.2
AL04	D0_Data(08)	8.8
AL05	$\overline{\text{Switch\_Clk\_A}}$	7.87
AL06	D0_Data(26)	8.5
AL07	D0_Data(19)	8.2
AL08	D0_Addr(07)	9.11
AL09	D0_Data(25)	7.7
AL10	D0_Addr(00)	9.01
AL11	D0_Addr(09)	6.61
AL12	D1_Data(02)	7.7
AL13	D1_Data(09)	6.6
AL14	D1_Addr(09)	8.81
AL15	D3_Data(06)	5.1

Table 224: Input Capacitance (pF) (Continued) (Page 9 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AL16	D1_WE	8.91
AL17	D3_Data(04)	7.5
AL18	D3_CS	8.91
AL19	D3_Addr(09)	9.11
AL20	D2_Data(05)	7.7
AL21	D2_Addr(09)	8.91
AL22	D2_CS	9.21
AL23	D6_Data(00)	7.9
AL24	D6_Data(07)	8.3
AL25	DA_$\overline{\text{Clk}}$	9.2
AL26	D6_Data(02)	8.4
AL27	D6_Addr(05)	9.8
AL28	D6_Addr(00)	10.2
AL29	D6_Addr(10)	10.3
AL30	D6_DQS(0)	9.5
AL31	D6_CS	11
AL32	PCI_Grant	12.35
AL33	PCI_Request	13.05
AM01	DE_BA(0)	11.31
AM03	D0_Data(05)	9.5
AM05	D0_Data(27)	9.4
AM07	D0_Addr(06)	9.91
AM09	D0_WE	9.71
AM11	D0_Addr(08)	9.51
AM13	D1_Data(12)	8
AM15	D1_Addr(03)	9.21
AM17	D3_Data(05)	8.2
AM19	D2_Data(12)	8
AM21	D2_Addr(03)	9.41
AM23	D6_Data(01)	8.6
AM25	DA_Clk	9.9
AM27	D6_Data(03)	9.3
AM29	D6_Parity(00)	10
AM31	D6_DQS(3)	10.2
AM33	PCI_IntA	13.05
AN01	D0_Data(06)	9.3

Table 224: Input Capacitance (pF) (Continued) (Page 10 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
AN02	D0_Data(04)	10.1
AN03	D0_Data(07)	9.9
AN04	D0_Data(17)	9.8
AN05	Switch_Clk_A	7.77
AN06	D0_Addr(03)	10.81
AN07	D0_Data(18)	9
AN08	D0_Data(24)	9.1
AN09	D0_CS	9.41
AN10	D0_Addr(01)	9.91
AN11	D1_Data(01)	7.4
AN12	D1_Data(10)	8.8
AN13	D1_Data(13)	6.4
AN14	D1_Addr(04)	9.71
AN15	D1_Addr(08)	9.91
AN16	D1_DQS(1)	8.5
AN17	D3_Addr(10)	9.91
AN18	D2_Data(00)	8.5
AN19	D2_Data(06)	8.8
AN20	D2_Data(11)	8.7
AN21	D2_Addr(02)	10.21
AN22	D2_Addr(08)	10.31
AN23	D6_Parity(01)	9
AN24	D6_Data(06)	9
AN25	D6_Data(11)	9.2
AN26	D6_Addr(01)	10.5
AN27	DA_CAS	10.5
AN28	D6_Data(05)	10.2
AN29	DA_BA(0)	11
AN30	D6_Addr(06)	11.4
AN31	D6_DQS(1)	10.6
AN32	D6_DQS_Par(00)	10.8
AN33	D6_DQS(2)	11.7
B01	SCH_Addr(16)	11.31
B03	SCH_Addr(13)	10.71
B05	SCH_Addr(01)	10.61
B07	D4_Addr(12)	9.91

Table 224: Input Capacitance (pF) (Continued) (Page 11 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
B09	SCH_Data(06)	9.71
B11	D4_Addr(07)	9.51
B13	DD_$\overline{\text{Clk}}$	9
B15	D4_Data(25)	8
B17	DS1_DQS(0)	8.2
B19	DS1_Data(13)	8
B21	DS1_Data(05)	8.2
B23	DS0_Addr(05)	9.81
B25	DS0_Data(29)	8.9
B27	DS0_Addr(03)	10.51
B29	DS0_Data(23)	10
B31	DS0_Data(02)	10.2
B33	Clock125	11.65
C01	SCH_Addr(17)	11.31
C02	SCH_Addr(14)	10.61
C03	SCH_Addr(09)	10.41
C04	SCH_Addr(12)	10.01
C05	Switch_Clk_B	7.87
C06	SCH_Data(12)	9.71
C07	SCH_Clk	9.41
C08	D4_Addr(09)	9.11
C09	SCH_Data(14)	8.91
C10	SCH_Data(01)	9.01
C11	D4_Addr(06)	8.81
C12	D4_Addr(00)	8.91
C13	DD_Clk	8.4
C14	D4_Data(17)	7.6
C15	D4_Data(03)	7.7
C16	D4_Data(10)	7.7
C17	DS1_WE	8.61
C18	DS1_Data(27)	7.7
C19	DS1_DQS(1)	7.9
C20	DS1_Data(21)	7.7
C21	DC_RAS	8.7
C22	DS0_Addr(11)	9.21
C23	DS0_Addr(06)	9.11

Table 224: Input Capacitance (pF) (Continued) (Page 12 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
C24	DS0_DQS(1)	8.3
C25	DS0_Data(21)	8.2
C26	DS0_Addr(04)	9.61
C27	DS0_Data(15)	8.8
C28	DS0_Data(22)	9.2
C29	DS0_Data(08)	9.3
C30	DS0_Data(04)	9.5
C31	DS0_Data(05)	10
C32	$\overline{\text{Operational}}$	10.95
C33	Core_Clock	11.65
D01	$\overline{\text{DASL\_In\_B(0)}}$	9.3
D03	SCH_Addr(15)	9.91
D05	SCH_Addr(10)	9.21
D07	SCH_Data(13)	9.31
D09	D4_Addr(08)	8.91
D11	D4_DQS(0)	7.2
D13	D4_Data(26)	7.2
D15	D4_Data(02)	7.3
D17	D4_Data(05)	7
D19	DS1_DQS(2)	7.5
D21	DS1_Data(12)	7.5
D23	DC_$\overline{\text{Clk}}$	8.6
D25	DC_BA(0)	9.3
D27	DS0_Data(26)	8.7
D29	DS0_Data(11)	8.8
D31	DMU_D(01)	10.25
D33	DMU_D(00)	11.55
E02	DASL_In_B(0)	8.5
E03	Spare_Tst_Rcvr(1)	7.81
E04	SCH_Addr(05)	9.21
E05	SCH_Addr(18)	8.91
E06	SCH_Addr(03)	9.81
E07	SCH_Addr(04)	8.61
E08	SCH_Data(09)	8.41
E09	D4_Data(18)	7.1
E10	D4_CS	8.11

Table 224: Input Capacitance (pF) (Continued) (Page 13 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
E11	D4_DQS(1)	6.9
E12	D4_Data(29)	6.8
E13	D4_Data(27)	6.8
E14	D4_Data(16)	6.7
E15	D4_Data(12)	6.7
E16	DS1_CS	7.61
E17	DS1_Addr(03)	7.81
E18	DS1_Data(20)	6.4
E19	DS1_Data(25)	6.8
E20	DS1_Data(22)	6.9
E21	DS1_Data(11)	7.1
E22	DS1_Data(08)	7.2
E23	DC_Clk	8.4
E24	DS0_Addr(12)	8.61
E25	DS0_DQS(3)	7.7
E26	DS0_Data(27)	7.9
E27	DS0_Data(12)	8.1
E28	DS0_Data(10)	9.4
E29	$\overline{\text{Blade\_Reset}}$	9.25
E30	DMU_D(04)	9.55
E31	DMU_D(29)	9.95
E32	DMU_D(12)	10.75
F01	$\overline{\text{DASL\_In\_B(2)}}$	8.7
F03	DASL_In_B(2)	7.9
F05	SCH_Addr(07)	8.61
F07	SCH_Addr(08)	8.91
F09	SCH_Data(02)	7.91
F11	D4_WE	7.51
F13	D4_Data(30)	6.2
F15	D4_Data(13)	6.2
F17	DS1_Addr(10)	7.31
F19	DS1_Data(24)	6.4
F21	DS1_Data(07)	6.5
F23	DS1_Data(04)	6.8
F25	DS0_DQS(0)	7.3
F27	DS0_Data(06)	8.6

Table 224: Input Capacitance (pF) (Continued) (Page 14 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
F29	DMU_D(07)	8.85
F31	DMU_D(06)	10.25
F33	DMU_D(05)	11.05
G02	DASL_In_B(3)	8
G03	Spare_Tst_Rcvr(5)	7.51
G04	$\overline{\text{DASL\_In\_B(3)}}$	7.3
G05	SCH_R_$\overline{\text{Wrt}}$	10.01
G06	SCH_Data(10)	8.81
G07	SCH_Data(11)	7.91
G08	SCH_Data(17)	7.81
G09	SCH_Data(05)	7.71
G10	D4_Addr(04)	7.71
G11	DD_CAS	7.81
G12	D4_Data(22)	6.4
G13	D4_Data(08)	6.5
G14	D4_Data(07)	6.5
G15	DS1_Addr(08)	7.61
G16	DS1_Addr(11)	7.61
G17	DS1_Addr(09)	7.41
G18	DS1_Addr(02)	7.71
G19	DS1_Addr(05)	7.71
G20	DS1_Data(30)	6
G21	DS1_Data(29)	6.1
G22	DS1_Data(15)	6.1
G23	DS1_Data(01)	6.4
G24	DS0_Addr(07)	8.41
G25	DS0_Data(30)	7.2
G26	DS0_Data(17)	7.3
G27	PCI_Bus_NM_Int	9.65
G28	DMU_D(02)	9.05
G29	DMU_D(11)	10.25
G30	DMU_D(10)	9.55
G31	DMU_D(30)	9.65
G32	DMU_D(08)	10.35
H01	$\overline{\text{DASL\_In\_B(1)}}$	8
H07	SCH_Addr(06)	7.91

Table 224: Input Capacitance (pF) (Continued) (Page 15 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
H09	D4_Data(06)	6.5
H11	D4_Addr(03)	7.51
H13	D4_Data(23)	6
H15	DS1_Addr(07)	7.31
H17	DS1_Addr(04)	7.11
H19	DS1_Addr(06)	7.41
H21	DS0_Data(07)	6.3
H23	DS0_Addr(08)	7.91
H25	DS0_Data(16)	6.5
H27	DMU_D(16)	8.15
H29	DMU_D(15)	8.95
H31	DMU_D(14)	9.75
H33	DMU_D(13)	10.45
J01	$\overline{\text{DASL\_In\_B(4)}}$	7.7
J02	DASL_In_B(1)	7.2
J03	DASL_In_B(4)	6.9
J06	MG_Data	7.88
J07	MG_Clk	7.28
J08	MG_nIntr	6.98
J10	SCH_Data(16)	7.11
J11	SCH_Data(03)	7.01
J12	D4_Addr(02)	6.91
J13	DD_BA(1)	6.91
J14	D4_Data(21)	5.7
J15	DS1_Data(17)	5.8
J16	DS1_Addr(12)	6.91
J17	DC_BA(1)	6.9
J18	DS1_Addr(01)	7.11
J19	DS1_Data(31)	6
J20	DS1_Data(18)	6
J21	DS1_Data(16)	6
J22	DS0_Addr(09)	7.21
J23	DS0_WE	7.31
J24	DS0_Data(18)	6.3
J25	DMU_D(25)	7.15
J26	DMU_D(24)	7.35

Table 224: Input Capacitance (pF) (Continued) (Page 16 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
J27	DMU_D(23)	8.35
J28	DMU_D(22)	8.85
J29	DMU_D(03)	8.65
J30	DMU_D(20)	9.15
J31	DMU_D(19)	9.35
J32	DMU_D(18)	9.65
J33	DMU_D(17)	10.15
K01	DASL_In_B(5)	7.6
K03	$\overline{\text{DASL\_In\_B(5)}}$	6.8
K07	Boot_Picocode	6.98
K13	DD_RAS	6.71
K15	D4_Data(09)	5.4
K19	DS1_Data(28)	5.6
K21	DS1_Data(02)	5.6
K23	DS0_Data(19)	5.5
K25	DMU_D(09)	7.15
K27	DMU_D(21)	8.05
K29	DMU_D(28)	8.55
K31	DMU_D(27)	9.25
K33	DMU_D(26)	10.05
L02	$\overline{\text{DASL\_In\_B(6)}}$	7
L03	DASL_In_B(6)	6.6
L04	Boot_PPC	7.68
L12	SCH_Data(04)	6.21
L13	D4_Addr(05)	6.31
L15	D4_Data(20)	5.1
L16	D4_Data(01)	5.1
L17	DS1_Data(10)	5.2
L18	DS1_Data(09)	5.2
L19	DS0_Data(25)	5.2
L20	DS1_Data(03)	5.3
L22	DS0_Data(31)	5.1
L23	DMU_C(10)	6.05
L24	DMU_C(09)	6.75
L25	DMU_C(08)	7.15
L26	DMU_C(07)	7.05

Table 224: Input Capacitance (pF) (Continued) (Page 17 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
L27	DMU_C(06)	7.25
L28	DMU_C(05)	7.85
L29	DMU_C(04)	8.25
L30	DMU_C(03)	8.95
L31	DMU_C(02)	9.05
L32	DMU_C(01)	9.45
L33	DMU_C(00)	9.95
M03	$\overline{\text{DASL\_In\_B(7)}}$	6.4
M07	PCI_Speed	6.78
M13	D4_Addr(10)	6.21
M15	D4_DQS(3)	4.9
M17	D4_Data(00)	4.9
M19	DS0_Addr(02)	6.31
M21	DS0_Data(24)	5
M23	DMU_C(16)	6.15
M25	DMU_C(15)	7.15
M27	DMU_C(14)	7.15
M29	DMU_C(13)	8.35
M31	DMU_C(12)	8.85
M33	DMU_C(11)	9.95
N04	DASL_In_B(7)	5.9
N14	D4_Addr(11)	6.11
N15	D4_DQS(2)	5
N16	D4_Data(15)	5.1
N17	DS1_Addr(00)	6.31
N18	DS1_Data(23)	5
N19	DC_CAS	6
N20	DS0_Addr(01)	6.31
N23	DMU_C(27)	6.15
N24	DMU_C(26)	6.55
N25	DMU_C(25)	7.05
N26	DMU_C(24)	7.25
N27	DMU_C(23)	7.15
N28	DMU_C(22)	7.55
N29	DMU_C(21)	7.85
N30	DMU_C(20)	8.45

Table 224: Input Capacitance (pF) (Continued) (Page 18 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
N31	DMU_C(19)	8.95
N32	DMU_C(18)	9.55
N33	DMU_C(17)	9.95
P15	D4_Data(28)	5.2
P19	DS0_Data(00)	5.3
P21	MC_Grant_B(1)	5.55
P23	DMU_B(02)	6.15
P25	DMU_B(01)	7.05
P27	DMU_B(00)	7.15
P29	DMU_C(30)	8.15
P31	DMU_C(29)	8.75
P33	DMU_C(28)	9.85
R04	LU_Addr(08)	8.01
R05	LU_Data(33)	7.51
R07	LU_Data(04)	7.21
R08	LU_Data(05)	6.81
R17	D4_Data(14)	5.6
R18	DS1_DQS(3)	5.6
R20	MC_Grant_B(0)	5.65
R21	Switch_BNA	5.55
R22	DMU_B(18)	5.85
R23	DMU_B(13)	6.15
R24	DMU_B(12)	6.55
R25	DMU_B(11)	6.85
R26	DMU_B(10)	7.15
R27	DMU_B(09)	6.95
R28	DMU_B(08)	7.55
R29	DMU_B(07)	8.15
R30	DMU_B(06)	8.55
R31	DMU_B(05)	8.95
R32	DMU_B(04)	9.45
R33	DMU_B(03)	9.95
T01	Spare_Tst_Rcvr(3)	7.52
T03	Spare_Tst_Rcvr(8)	6.61
T07	LU_Data(03)	7.21
T09	LU_Data(02)	6.61

Table 224: Input Capacitance (pF) (Continued) (Page 19 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
T15	LU_Data(24)	7.21
T19	Send_Grant_B	5.95
T21	RES_Data	5.55
T23	DMU_B(19)	6.15
T25	$\overline{\text{JTAG\_TRst}}$	6.85
T27	DMU_B(17)	6.95
T29	DMU_B(16)	7.65
T31	DMU_B(15)	8.95
T33	DMU_B(14)	9.75
U01	LU_Data(30)	9.41
U03	LU_Data(35)	8.11
U05	Spare_Tst_Rcvr(0)	5.47
U06	Testmode(1)	5.78
U08	LU_Data(08)	6.61
U10	LU_Data(34)	6.21
U12	LU_Data(11)	6.11
U13	LU_Data(01)	6.41
U15	LU_Data(00)	7.11
U19	MGrant_A(1)	5.95
U21	RES_Sync	5.55
U22	JTAG_TMS	5.75
U23	Rx_LByte(0)	6.15
U24	DMU_B(29)	6.55
U25	DMU_B(28)	6.85
U26	DMU_B(27)	6.95
U27	DMU_B(26)	7.25
U28	DMU_B(25)	7.25
U29	DMU_B(24)	7.85
U30	DMU_B(23)	8.15
U31	DMU_B(22)	8.75
U32	DMU_B(21)	9.45
U33	Spare_Tst_Rcvr(9)	8.12
V01	Spare_Tst_Rcvr(6)	7.51
V03	Spare_Tst_Rcvr(7)	6.58
V05	Testmode(0)	5.98
V07	LU_Data(09)	7.21

Table 224: Input Capacitance (pF) (Continued) (Page 20 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
V09	LU_Data(10)	6.61
V11	LU_Data(14)	5.91
V13	LU_Data(18)	6.41
V15	LU_Data(12)	7.21
V19	MGrant_A(0)	5.95
V21	I_FreeQ_Th	5.55
V23	Rx_LByte(1)	6.15
V25	DMU_B(20)	6.85
V27	DMU_A(02)	6.95
V29	DMU_A(01)	7.65
V31	DMU_A(00)	8.95
V33	DMU_B(30)	9.75
W01	LU_Data(20)	9.41
W04	LU_Data(13)	8.01
W05	LU_Data(07)	7.51
W06	LU_Data(06)	7.71
W07	LU_Data(15)	7.21
W08	LU_Data(16)	6.81
W09	LU_Data(21)	6.51
W10	LU_Data(25)	6.31
W11	LU_Data(31)	5.81
W12	LU_Data(26)	6.01
W13	LU_Data(19)	6.41
W14	LU_Data(27)	6.81
W16	D1_Addr(10)	6.91
W17	D3_Data(08)	5.7
W18	D2_Data(02)	5.6
W20	Send_Grant_A	5.65
W21	MC_Grant_A(1)	5.55
W22	JTAG_TDI	5.85
W23	DMU_A(13)	6.15
W24	DMU_A(12)	6.55
W25	DMU_A(11)	6.85
W26	DMU_A(10)	7.15
W27	DMU_A(09)	6.95
W28	DMU_A(08)	7.55

Table 224: Input Capacitance (pF) (Continued) (Page 21 of 21) T_A = 25 °C, f = 1MHz, V_{DD}= 3.3V ±0.3V)

Grid Position	Signal Name	Total InCap
W29	DMU_A(07)	8.15
W30	DMU_A(06)	8.55
W31	DMU_A(05)	8.95
W32	DMU_A(04)	9.45
W33	DMU_A(03)	9.95
Y03	$\overline{\text{DASL\_In\_A(7)}}$	6.2
Y05	LU_Data(32)	7.51
Y07	LU_Data(17)	7.41
Y09	LU_Data(22)	6.81
Y11	LU_Addr(01)	5.81
Y15	D1_Data(05)	5.2
Y19	D2_Data(15)	5.3
Y21	MC_Grant_A(0)	5.55
Y23	DMU_A(19)	6.15
Y25	DMU_A(18)	6.95
Y27	DMU_A(17)	7.15
Y29	DMU_A(16)	8.15
Y31	DMU_A(15)	8.75
Y33	DMU_A(14)	9.85

Table 225: Operating Supply Voltages

Symbol	Parameter	Rating			Units	Notes
		Min	Typ	Max		
V_{DD25} V_{DD}	2.5 V Power Supply	2.375	2.5	2.625	V	1
V_{DD33} V_{DD2} V_{DD3} V_{DD5}	3.3 V Power Supply	3.135	3.3	3.465	V	1
V_{DD18} V_{DD4}	1.8 V Power Supply	1.71	1.8	1.89	V	1
PLLA_V_{DD} PLLB_V_{DD} PLLC_V_{DD}	PLL voltage reference	1.71	1.8	1.89	V	1, 1
$V_{REFR1(2-0)}$ $V_{REFR2(8-0)}$	SSTL2 Power Supply (used for SSTL2 I/O)	1.1875	1.25	1.3125	V	1

1. Important Power Sequencing Requirements: (the following conditions must be met at all times, including power-up and power-down:
   V_{REF}*(1.25 V reference) ≤ V_{DD25} +0.4 V
   PLL*_V_{DD}(2.5 V reference) ≤ V_{DD33} +0.4 V
   V_{DD15} ≤ V_{DD25} + 0.4 V
   V_{DD25} ≤ V_{DD33} +0.4 V
   V_{DD33} ≤ V_{DD25} + 1.9 V
1. See also *PLL Filter Circuit* on page 65.

Table 226: Thermal Characteristics

Thermal Characteristic	Min	Nominal	Max	Units
Estimated Power Dissipation	TBD	TBD	TBD	°C
Operating Junction Temperature (Tj)	0		105	°C

14.1 Driver Specifications

Table 227: Definition of Terms

Term	Definition
MAUL	Maximum allowable up level. The maximum voltage that can be applied without affecting the specified reliability. Cell functionality is not implied. Maximum allowable applies to overshoot only.
MPUL	Maximum positive up level. The most positive voltage that maintains cell functionality. The maximum positive logic level.
LPUL	Least positive up level. The least positive voltage that maintains cell functionality. The minimum positive logic level.
MPDL	Most positive down level. The most positive voltage that maintains cell functionality. The maximum negative logic level.
LPDL	Least positive down level. The least positive voltage that maintains cell functionality. The minimum negative logic level.
MADL	Minimum allowable down level. The minimum voltage that can be applied without affecting the specified reliability. Minimum allowable applies to undershoot only. Cell functionality is not implied.

Table 228: 1.8 V CMOS Driver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
CMOS	V_{DD}[3] + 0.45	V_{DD}[3]	V_{DD}[3] - 0.45	0.45	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. V_{DD} ranges as specified in *Table 225* (Typical = 1.8 V).

Table 229: 1.8 V CMOS Driver Minimum DC Currents at Rated Voltage V_{DD} = 1.65 V, T = 100°C

Driver Type	V_{HIGH} (V)	I_{HIGH} (mA)	V_{LOW} (V)	I_{LOW} (mA)
CMOS 50 ohm driver outputs	1.2	8.0/23.0[1]	0.45	7.8[1]

1. 23 mA is the electromigration limit for 100K power on hours (POH) = 100°C and 100% duty cycle. This limit can be adjusted for different temperature, duty cycle, and POH. Consult your IBM application engineer for further details.

Table 230: 2.5 V CMOS Driver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
CMOS	V_{DD}[3] + 0.6	V_{DD}[3]	2.0	0.4	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. V_{DD} ranges as specified in *Table 225* (Typical = 2.5 V).

Table 231: 2.5 V CMOS Driver Minimum DC Currents at Rated Voltage V_{DD} = 2.3 V, T = 100°C

Driver Type	V_{HIGH} (V)	I_{HIGH} (mA)	V_{LOW} (V)	I_{LOW} (mA)
CMOS 50 ohm driver outputs	2.0	5.2/23[1]	0.4	6.9[1]

1. 23 mA is the electromigration limit for 100K power on hours (POH) = 100°C and 100% duty cycle. This limit can be adjusted for different temperature, duty cycle, and POH. Consult your IBM application engineer for further details.

Table 232: 3.3 V-Tolerant 2.5 V CMOS Driver DC Voltage Specifications (see note 1)

Function	MAUL (V)[2]	MPUL (V)[3]	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[4]
LVTTL	3.9	V_{DD}[5]	2.0	0.4	0.00	-0.60

1. All levels adhere to the JEDEC Standard JESD12-6, “Interface Standard for Semi-Custom Integrated Circuits,” March 1991.
2. Maximum allowable applies to overshoot only. Output disabled.
3. Output active.
4. Minimum allowable applies to undershoot only.
5. V_{DD} ranges as specified in *Table 225* (Typical = 2.5 V).

Table 233: 3.3 V LVTTL Driver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
LVTTL	V_{DD330}[3] + 0.3	V_{DD330}[3]	2.4	0.4	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. $V_{DD\ 33}$ ranges as specified in *Table 225* (Typical = 3.3 V).

Table 234: 3.3 V LVTTL/5.0 V-Tolerant Driver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
LVTTL	V_{DD330}[3] + 0.3	V_{DD330}[3]	2.4	0.4	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. $V_{DD\ 33}$ ranges as specified in *Table 225* (Typical = 3.3 V).

Table 235: 3.3 V LVTTL Driver Minimum DC Currents at Rated Voltage (V_{DD} = 3.0 V, T = 100 °C)

Driver Type	V_{HIGH} (V)	I_{HIGH} (mA)	V_{LOW} (V)	I_{LOW} (mA)
LVTTL 50 ohm driver outputs	2.40	10.3/23[1]	0.4	7.1[1]

1. 23 mA is the electromigration limit for 100K power on hours (POH) = 100°C and 100% duty cycle. This limit can be adjusted for different temperature, duty cycle, and POH. Consult your IBM application engineer for further details.

14.2 Receiver Specifications

Table 236: 1.8 V CMOS Receiver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
CMOS	V_{DD}[3] + 0.45	V_{DD}[3]	0.65V_{DD}[3]	0.35V_{DD}[3]	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. V_{DD} ranges as specified in *Table 225* (Typical = 1.8 V).

Table 237: 2.5 V CMOS Receiver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
CMOS	V_{DD}[3] + 0.6	V_{DD}	1.7	0.70	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. V_{DD} ranges as specified in *Table 225* (Typical = 2.5 V).

Table 238: 3.3 V LVTTL Receiver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
LVTTL	V_{DD330}[3] + 0.3	V_{DD330}[3]	2.00	0.80	0.00	-0.60

1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.
3. $V_{DD\ 33}$ ranges as specified in *Table 225* (Typical = 3.3 V).

Table 239: 3.3 V LVTTL/5V-Tolerant Receiver DC Voltage Specifications

Function	MAUL (V)[1]	MPUL (V)	LPUL (V)	MPDL (V)	LPDL (V)	MADL (V)[2]
LVTTL	5.5 V	5.5 V	2.00	0.80	0.00	-0.60

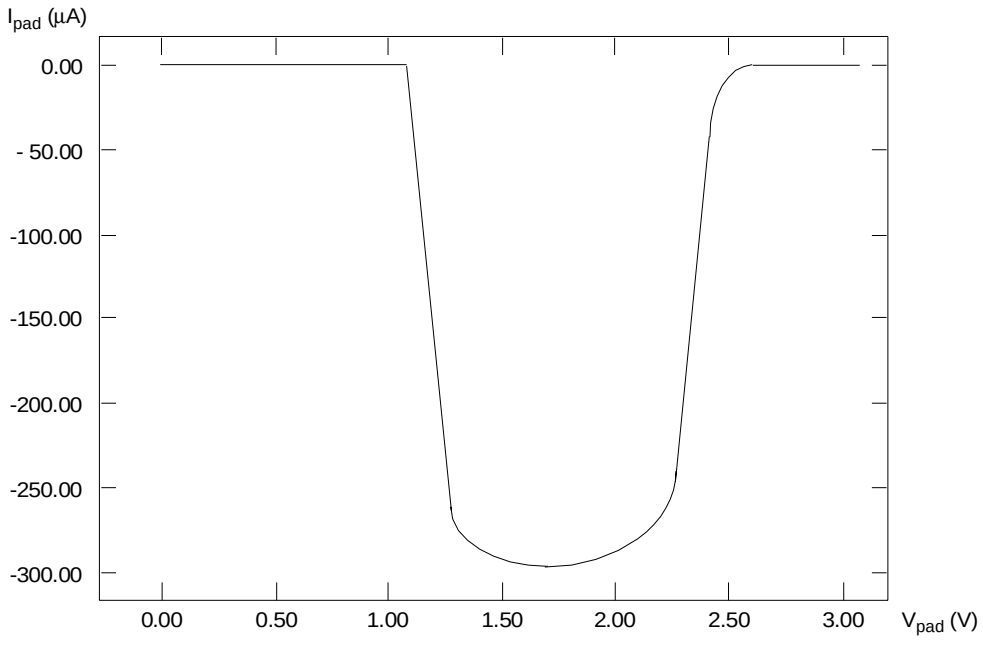
1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.

Table 240: Receiver Maximum Input Leakage DC Current Input Specifications

Function	I_{IL} (µA)	I_{IH} (µA)
Without pull-up element or pull-down element	0 at V_{IN} = LPDL	0 at V_{IN} = MPUL
With pull-down element	0 at V_{IN} = LPDL	200 at V_{IN} = MPUL
With pull-up element	-150 at V_{IN} = LPDL	0 at V_{IN} = MPUL

1. See section *3.3V LVTTL/5V Tolerant BP33 and IP33 Receiver Input Current/Voltage Curve on page 429*.

Figure 75: 3.3V LVTTL/5V Tolerant BP33 and IP33 Receiver Input Current/Voltage Curve



1. Curve shows best case process - 0°C, 3.6V

14.3 Other Driver and Receiver Specifications

Table 241: LVDS Receiver DC Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
V_{DD}	Device supply voltage	1.65	1.8	1.95	V	Receiver uses only V_{DD} supply.
Temp	Temperature Range	0	50	100	°C	
Rec Pwr	Input buffer power			9.3	mW	Including on-chip terminator V_{PAD} - V_{PADN} = 0.4 V
V_{IH}	Receiver input voltage			V_{DD} + 0.20	V	Receiver ESD connected to V_{DD}
V_{IL}	Receiver input voltage	-0.20			V	
V_{IH} - V_{IL}	Receiver input voltage range	100			mV	@600 MHz
V_{ICM}	Receiver common mode range	0	1.25	V_{DD}	V	

Notes:

1. All DC characteristics are based on power supply and temperature ranges as specified above.
2. LVDS design reference: IEEE Standard for Low-Voltage Differential Signals (LVDS) for Scalable Coherent Interface (SCI), IEEE Standard 1596.3,1996.
3. Maximum frequency is load and package dependent. 600 MHz (1.2 Gbps) is achievable with a minimum of 100 mV input swing over the wide common range as specified. The customer is responsible for determining optimal frequency and switching capabilities through thorough simulation and analysis.

Table 242: SSTL2 DC Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
V_{DD}	Device supply voltage	1.65	1.8	1.95	V	
V_{DDQ}	Output supply voltage	2.3	2.5	2.7	V	$V_{DDQ} = V_{DD250}$
V_{TT}	Termination voltage	1.11 - 1.19	1.25	1.31 - 1.39	V	$0.5*V_{DDQ}$
V_{REF}	Differential input reference voltage	1.15	1.25	1.35	V	$0.5*V_{DDQ}$
V_{OH} (Class II)	Output high voltage	1.95			V	I_{OH} = 15.2 mA @ 1.95V
V_{OL} (Class II)	Output low voltage			0.55	V	I_{OL} = 15.2 mA @ 0.55V
R_{OH} max (Class II)	Max pull-up impedance			36.2	Ω	

Notes:

1. All SSTL2 specifications are consistent with JEDEC committee re-ballot (JC-16-97-58A), 10/14/97.
2. Di/dt and performance are chosen by performance level selection (A and B).
   a. Performance level A is targeted to run at 200 MHz or faster depending on loading conditions. Di/dt is comparable to 110 mA/ns 2.5 V/3.3 V LVTTL driver.
   b. Performance level B is targeted to run at 250 MHz or faster depending on loading conditions. Di/dt is comparable to 150 mA/ns 2.5 V/3.3 V LVTTL driver.
3. The differential input reference supply (V_{REF}) is brought on chip through VSSTL2R1 and VSSTL2R2 I/O cells.
4. Termination voltage (V_{TT}) is generated off chip.
5. SSTL2 driver is rated at 20 mA @100°C and 50% duty cycle for 100k power on hours (POH).

Table 242: SSTL2 DC Specifications

Symbol	Parameter	Min	Nom	Max	Units	Comments
R_{OL} max (Class II)	Max pull-down impedance			36.2	Ω	
V_{IH}	Input high voltage	V_{REF} + 0.18		V_{DDQ} + 0.3	V	
V_{IL}	Input low voltage	-0.3		V_{REF} - 0.18	V	
I_{OZ}	3-state leakage current	0		10	µA	Driver Hi-Z
Temp	Temperature	0	50	100	°C	

Notes:

1. All SSTL2 specifications are consistent with JEDEC committee re-ballot (JC-16-97-58A), 10/14/97.
2. Di/dt and performance are chosen by performance level selection (A and B).
   a. Performance level A is targeted to run at 200 MHz or faster depending on loading conditions. Di/dt is comparable to 110 mA/ns 2.5 V/3.3 V LVTTL driver.
   b. Performance level B is targeted to run at 250 MHz or faster depending on loading conditions. Di/dt is comparable to 150 mA/ns 2.5 V/3.3 V LVTTL driver.
3. The differential input reference supply (V_{REF}) is brought on chip through VSSTL2R1 and VSSTL2R2 I/O cells.
4. Termination voltage (V_{TT}) is generated off chip.
5. SSTL2 driver is rated at 20 mA @100°C and 50% duty cycle for 100k power on hours (POH).

15. Glossary of Terms and Abbreviations

Term	Definition
ALU	arithmetic and logic unit
API	application programming interface
ARB	arbitration
ARDL	advance rope with optional delete leaf
ARP	Address Resolution Protocol
ATH	actual threshold
BCB	buffer control block
BCI	byte count information
BEB	binary exponential back-off
BFQ	Buffer Free Queue
BGP	Border Gateway Protocol
bird	An intermediate leaf. It occurs when the PSCBLine contains an intermediate LCBA pointing to this leaf and an NPA pointing to the next PSCB.
BL	burst length
BSM-CCGA	bottom surface metallurgy - ceramic column grid array
byte	8 bits
CAB	Control Access Bus
CHAP	Challenge-Handshake Authentication Protocol
CIA	Common Instruction Address
CLNS	connectionless-mode network service
CLP	See Core Language Processor
Core Language Processor (CLP)	The picoprocessor core, also referred to as coprocessor 0. The CLP executes the base instruction set and controls thread swapping and instruction fetching.
CP	control point
CPF	control point function
CPIX	Common Programming Interface
CRC	see *cyclic redundancy check*
CS	Control Store

Term	Definition
CSA	Control Store Arbiter
cyclic redundancy check (CRC)	A system of error checking performed at both the sending and receiving station after a block-check character has been accumulated.
DA	destination address
DASL	data-aligned synchronous link
data store	In the Network Processor, the place where a frame is stored while waiting for processing or forwarding to its destination.
DBGS	Debug Services
DDR	double data rate
DiffServ	Differentiated Services
Distinguishing Position (DISTPOS)	The index value of a first mismatch bit between the input key and the reference key found in a leaf pattern
DISTPOS	See *Distinguishing Position (DISTPOS)*
DLQ	Discard List Queue
DMU	Data Mover Unit
doubleword	2 words
DPPU	dyadic protocol processor unit
DQ	Discard Queue
DRAM	dynamic random-access memory
DS	see *data store*
DSCP	DiffServ code point
DSI	Distributed Software Interface
DT	Direct Table
DVMRP	Distance Vector Multicast Routing Protocol
E-	Egress
ECC	Error correcting code
ECMP	equal-cost multipath
EDS	Enqueuer / Dequeuer / Scheduler
E-DS	Egress Data Store

Term	Definition
E-GDQ	Discard Queue Stack. Holds frames that need to be discarded. This is used by the hardware to discard frames when the egress DS is congested or to re-walk a frame marked for discard for a half duplex port. Used by the picocode to discard frames that do not have header twins allocated.
Egress EDS	Egress Enqueuer / Dequeuer / Scheduler
Enet MAC	Ethernet MAC
EOF	end of frame
EPC	Embedded Processor Complex
E-PMM	Egress-Physical MAC Multiplexer
EWMA	exponentially weighted moving average
exponentially weighted average	See *exponentially weighted moving average*
exponentially weighted moving average	A method of smoothing a sequence of instantaneous measurements. Typically, the average A(t) at time t is combined with the measurement M(t) at time t to yield the next average value: A(t+Dt) = W*M(t)+(1-W)*A(t) Here weight W is a number with 0 < W <= 1. If the weight is 1, then the average is just the previous value of the measurement and no smoothing occurs. Else previous values of M contribute to the current value of A with more recent M values being more influential.
FCB	frame control block
FFA	flexible frame alternation
FHE	frame header extension
FHF	Frame Header Format
FM	full match
FPGA	field-programmable gate array
FTA	first twin-buffer address
GDH	See *General Data Handler*.
General Data Handler (GDH)	A type of thread used to forward frames in the EPC. There are 28 GDH threads.
General PowerPC Handler Request (GPH-Req)	A type of thread in the EPC that processes frames bound to the embedded PowerPC. Work for this thread is usually the result of a reenqueue action to the PPC queue (it processes data frames when there are no entries to process in the PPC queue).

Term	Definition
General PowerPC Handler Response (GPH-Resp)	A type of thread in the EPC that processes responses from the embedded PowerPC. Work for this thread is dispatched due to an interrupt and does not use dispatcher memory.
General Table Handler (GTH)	The GTH executes commands not available to a GDH or GFH thread, including hardware assist to perform tree inserts, tree deletes, tree aging, and rope management. It processes data frames when there are no tree management functions to perform.
GFH	See *Guided Frame Handler*.
GFQ	Guided Frame Queue
GMII	Gigabit Medium Independent Interface
GPH-Req	See *General PowerPC Handler Request*.
GPH-Resp	See *General PowerPC Handler Response*.
GPP	general-purpose processor
GPQ	PowerPC queue. The queue that contains frames re-enqueued for delivery to the GPH for processing.
GPR	general-purpose register
GTH	See *General Table Handler*.
GUI	graphical user interface
Guided Frame Handler (GFH)	There is one GFH thread available in the EPC. A guided frame can be processed only by the GFH thread, but it can be configured to enable it to process data frames like a GDH thread. The GFH executes guided frame-related picocode, runs chip management-related picocode, and exchanges control information with a control point function or a remote network processor. When there is no such task to perform and the option is enabled, the GFH can execute frame forwarding-related picocode.
GxH	Generalized reference to any of the defined threads of the EPC
halfword	2 bytes
HC	Hardware Classifier
HDLC	See *high-level data link control*.
high-level data link control (HDLC)	In data communication, the use of a specified series of bits to control data links in accordance with the International Standards for HDLC: ISO 3309 Frame Structure and ISO 4335 Elements of Procedures.
I-	Ingress
ICMP	Internet Control Message Protocol

Term	Definition
I-DS	Ingress Data Store
Ingress EDS	Ingress Enqueuer / Dequeuer / Scheduler
Ingress GDQ	Ingress General Data Queue
IMS	Interface Manager Services
IPC	interprocess communication
I-PMM	Ingress-Physical MAC Multiplexer
IPPS	Internet Protocol Version 4 Protocol Services
IPv4	Internet Protocol Version 4
I-SDM	Ingress Switch Data Mover
K	2^{10}, or 1024 in decimal notation
LCBA	leaf control block address
LDP	Label Distribution Protocol
leaf	A control block that contains the corresponding key as a reference pattern and other user data such as target blade number, QoS, and so on.
LH	latched high (a characteristic of a register or a bit in a register (facility)). When an operational condition sets the facility to a value of 1, the changes to the operational condition do not affect the state of the facility until the facility is read.
LID	lookup identifier
LL	latched low (a characteristic of a register or a bit in a register (facility)). When an operational condition sets the facility to a value of 0, the changes to the operational condition do not affect the state of the facility until the facility is read.
LLS	low-latency sustainable bandwidth
LPM	longest prefix match
LSb	least significant bit
LSB	least significant byte
LSP	label-switched path
LuDef	lookup definition
M	2^{20}, or 1 048 576 in decimal notation
MAC	medium access control

Term	Definition
Management Information Base (MIB)	In OSI, the conceptual repository of management information within an open system.
maximum burst size (MBS)	In the Network Processor Egress Scheduler, the duration a flow can exceed its guaranteed minimum bandwidth before it is constrained to its guaranteed minimum bandwidth.
maximum transmission unit (MTU)	In LANs, the largest possible unit of data that can be sent on a given physical medium in a single frame. For example, the MTU for Ethernet is 1500 bytes.
MBS	see *maximum burst size*
MCC	multicast count
MCCA	multicast count address
MH	MID handler
MIB	see *Management Information Base*
MID	Multicast ID
MM	MID Manager
MMS	MID Manager Services
MPLS	Multiprotocol Label Switching
Mpps	million packets per second
MSb	most significant bit
MSB	most significant byte
MSC	Message Sequence Charts
MTU	see *maximum transmission unit*
My_TB	My Target Blade
NBT	next bit to test
NFA	next frame control block (FCB) address
NLA	next leaf address
NLS	normal latency sustainable bandwidth
NP	Network Processor
NPA	next PCSB address pointer
NPASM	IBM Network Processor Assembler

Term	Definition
NPDD	Network Processor Device Driver
NPDDIS	Network Processor Device Driver Initialization Services
NPMS	Network Processor Manager Services
NPScope	IBM Network Processor Debugger
NPSIM	IBM Network Processor Simulator
NPTest	IBM Network Processor Test Case Generator
ns	nanosecond
NTA	next twin-buffer address
OQG	output queue grant
PAP	Password Authentication Protocol
PBS	peak bandwidth service
PC	program counter
PCB	Port Control Block
PCS	Physical Coding Sublayer
PCT	port configuration table
PHY	see *physical layer*. May also refer to a physical layer device.
physical layer	In the Open Systems Interconnection reference model, the layer that provides the mechanical, electrical, functional, and procedural means to establish, maintain, and release physical connections over the transmission medium. (T)
PLB	Processor Local Bus
PLL	phased lock loop
PMA	physical medium attachment
PMM	Physical MAC Multiplexer
PolCB	Policing Control Block
POS	packet over SONET
POST	power-on self-test
postcondition	An action or series of actions that the user program or the NPDD must perform after the function has been called and completed. For example, when the function that defines a table in the NPDD has been completed, the NPDD must dispatch a guided frame from the PowerPC core to instruct one or more EPCs to define the table.

Term	Definition
PPC	PowerPC
PPP	Point-to-Point Protocol
PPrev	Previous Discard Probability
precondition	A requirement that must be met before the user program calls the API function. For example, a precondition exists if the user program must call one function and allow it to be completed before a second function is called. One function that has a precondition is the function that deregisters the user program. The user program must call the register function to obtain a *user_handle* before calling the deregistering function.
PSCB	pattern search control block
PTL	Physical Transport Layer
PTS	Physical Transport Services
quadword	4 words
quality of service (QoS)	For a network connection, a set of communication characteristics such as end-to-end delay, jitter, and packet loss ratio.
QCB	queue control block
QoS	see *quality of service*
Rbuf	raw buffer
RCB	Reassembly Control Block
RCLR	read current leaf from rope
RED	random early detection
RMON	Remote Network Monitoring
RPC	remote procedure call
RTP	Real-Time Transport Protocol
RUM	Access type “reset under mask”
R/W	Access type “Read/Write”
RWR	Access type “Read with Reset”
SA	source address
SAP	service access point
SC	self clearing (a characteristic of a register or a bit in a register (facility)). When written to a value of 1, will automatically reset to a value of 0 when the indicated operation completes.

Term	Definition
SCB	schedule control block
SCI	Switch Cell Interface
SDC	shared DASL controller
SDM	Switch Data Mover
SMT	software-managed tree
SMII	Serial Media-Independent Interface
SOF	start-of-frame
SONET	Synchronous Optical Network
SPM	Serial/Parallel Manager
SRAM	static random-access memory
SS	System Services
SUM	Access type “set under mask”
SWI	Switch Interface
Target Blade Grant (TBG)	An internal facility used by the ingress scheduler to determine which target blades are accepting data transfer. Derived from the output queue grant (OQG) information the network processor receives.
TB	target blade
TBG	See *Target Blade Grant*.
TBR	target blade running
TB_SOF	target blade start-of-frame
TCP	Transmission Control Protocol
thread	A stream of picocode instructions that utilizes a private set of registers and shared computational resources within a DPPU. In the network processor, a DPPU provides both shared and private resources for four threads. Two threads are bound to a single picoprocessor, allowing for concurrent execution of two threads within a DPPU. The coprocessors are designed to support all four threads. In general, the computational resources (ALU, hardware-assist state machines, and so on) are shared among the threads. The private resources provided by each picoprocessor or coprocessor include the register set that makes up its architecture (GPRs, program counters, link stacks, Data Pool, and so on).
TLIR	tree leaf insert rope

Term	Definition
TLV	type length vectors
TMS	Table Management Services
TP	target port
TSDQFL	Tree Search Dequeue Free List
TS	Transport Services
TSE	Tree Search Engine
TSENQFL	Tree Search Enqueue Free List
TSFL_ID	tree search free list identifier
TSR	tree search result
TTL	time to live
UC	Unicast
UDP	User Datagram Protocol
WFQ	weighted fair queueing
word	4 bytes
ZBT	zero bus turnaround

Revision Log

Rev	Description of Modification
11/11/99	Initial release of databook for IBM32NPR161EPXCAC133 (revision 00), follow-up document to datasheet for IBMNPR100EXXCAB133 and IBM32NPR161EPXCAC133 (revision 01, 11/08/99)
09/25/00	Release IBM32NPR161EPXCAC133 Databook (revision 01), including revisions to all sections.