IBM PowerNP™
NP2G
Network Processor

Preliminary

February 12, 2003

IBM	IBM Logo
PowerPC	PowerNP

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

Note: This document contains information on products in the 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.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN “AS IS” BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
2070 Route 52, Bldg. 330
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.ibm.com/chips

np2_ds_title.fm.01
February 12, 2003

Contents

List of Tables

List of Figures

About This Book

This datasheet describes the IBM PowerNP NP2G and explains the basics of building a system using it.

A terms and abbreviations list is provided in *Section 15. Glossary of Terms and Abbreviations* on page 529.

Who Should Read This Manual

This datasheet provides information for network hardware engineers and programmers using the NP2G to develop interconnect solutions for Internet or enterprise network providers. It includes an overview of data flow through the device and descriptions of each functional block. In addition, it provides electrical, physical, thermal, and configuration information about the device.

Related Publications

IBM PowerPC 405GP Embedded Processor User's Manual
(http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405GP_Embedded_Processor)

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 in the following sections 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.
   - *Section 2. Physical Description*
   - *Section 3. Physical MAC Multiplexer*
   - *Section 4. Ingress Enqueuer / Dequeuer / Scheduler*
   - *Section 5. Ingress-to-Egress Wrap*
   - *Section 6. Egress Enqueuer / Dequeuer / Scheduler*
   - *Section 7. Embedded Processor Complex*
   - *Section 9. Serial / Parallel Manager Interface*
2. The bit notation in *Section 8. Tree Search Engine* and *Section 10. Embedded PowerPC™ Subsystem* 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.
3. Nibble numbering is the same as byte numbering. The left-most nibble is most significant and starts at zero.
4. All counters wrap back to zero when they exceed their maximum values. Exceptions to this rule are noted in the counter definitions.
5. Overbars ($\overline{\text{TxEnb}}$, for example) designate signals that are asserted "low."

6. 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
7. 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
8. For signal and field definitions, when a field is designated as “Reserved”:
   - It must be sent as zero as an input into the NP2G, either as a signal I/O or a value in a reserved field of a control block used as input to a picocode process.
   - It must not be checked or modified as an output from the NP2G, either as a signal I/O or a value in a reserved field of a control block used as input to an external code process.
   - 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.
- Six dyadic protocol processor units (DPPUs)
  - Two picocode processors per DPPU
  - 10 shared coprocessors per DPPU
  - Four threads per DPPU
  - Zero context switching overhead between threads
- Embedded PowerPC™ processor and external 33/66 MHz 32-bit PCI Bus for enhanced design flexibility.
  - supports RISCWatch through the JTAG interface
- Integrated Ethernet and packet over SONET (POS) medium access controls (MACs)
  - Up to two Gigabit Ethernet (plus a gigabit uplink/control point attachment port) or 20 Fast Ethernet ports, accessed through Serial Media-Independent (SMII), Gigabit Media-Independent (GMII), and Ten-Bit (TBI) interfaces, that support industry standard physical layer devices
  - 36 Ethernet statistics counters per MAC
  - Up to one million software-defined, hardware-assisted counters, enabling support of many standard Management Information Bases (MIBs) at wire speed
  - 8 OC-3c, 2 OC-12, or 2 OC-12c (plus an OC-12 or OC-12c uplink/control point attachment port) integrated POS interfaces that support industry standard POS framers
  - Hardware VLAN support (detection, tag insertion and deletion).
- Advanced flow control mechanisms that tolerate high rates of temporary oversubscription without TCP collapse.
- Fast lookups and powerful search engines based on geometric hash functions that yield lower collision rates than conventional bit-scrambling methods.
- Hardware support for port mirroring. Mirrored traffic can share bandwidth with user traffic or use a separate data path, eliminating the normal penalty for port mirroring.
- Support for jumbo frames.
  - Maximum length determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* on page 489).

  The value is increased by four octets for VLAN tagged frames.
- Hardware-managed, software-configured bandwidth allocation control of 2047 concurrent communication flows.
- 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
- 1088-pin Bottom Surface Metallurgy - Ceramic Column Grid Array (BSM-CCGA) package with 815 Signal I/O.
- IEEE® 1149.1a JTAG compliant.

1.2 Ordering Information

Part Number	Description
IBM32NP160EPXCAA133	IBM PowerNP NP2GA (R1.0) Network Processor
IBM32NP160EPXCAB133	IBM PowerNP NP2GB (R2.0) Network Processor

1.3 Overview

The IBM PowerNP™ NP2G Network Processor enables network hardware designers to create fast, powerful, and scalable systems. The NP2G contains an Embedded Processor Complex (EPC) in which processors and coprocessors work with hardware accelerators to increase processing speed and power. Additional features, such as integrated search engines, variable packet length schedulers, and support for QoS functions, support the needs of customers who require high function, high capacity, media-rate switching.

The EPC is the heart of the NP2G, evaluating, defining, and processing data. It maximizes the speed and processing power of the device and provides it with functionality above that of an independent switching device. Within the EPC, six dyadic protocol processor units (DPPUs) combine picocode processors, coprocessors, and hardware accelerators to support functions such as high-speed pattern search, data manipulation, internal chip management, frame parsing, and data prefetching.

The NP2G provides fast switching by integrating switching engine, search engine, and security functions on one device. It supports Layer 2 and 3 Ethernet frame switching, and includes three priority levels for port mirroring, high priority user frames, and low priority frames. It supports Ethernet, packet over SONET (POS), and Point-to-Point Protocol (PPP) protocols. Because of the device's ability to enforce hundreds of rules with complex range and action specifications, NP2G-based systems are uniquely suited for server clusters.

Systems developed with the NP2G use a distributed software model. 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, all of which can decrease the time to market for new applications.

In order to take advantage of these features, a designer must know how the device works and how it fits into a system. The following sections discuss the basic placement of the device within a system, its major functional blocks, and the movement of data through it. The chapters following this overview explore all of these issues in detail.

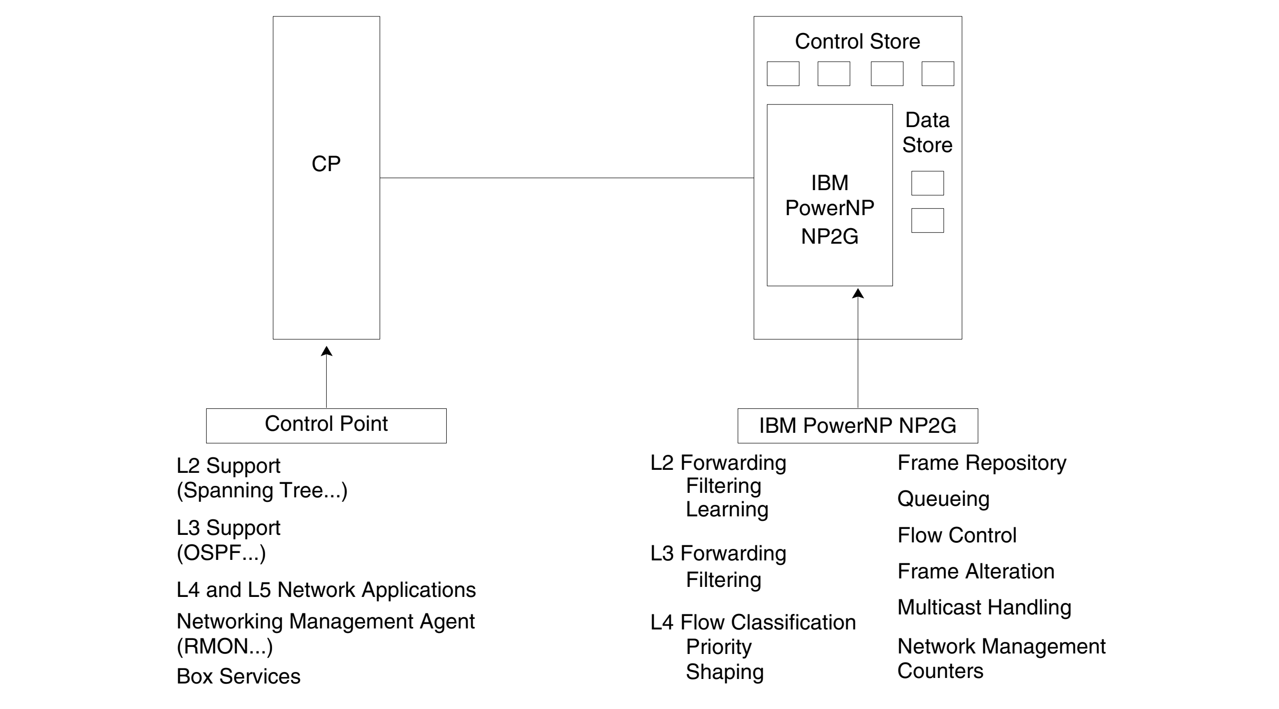
1.4 NP2G-Based Systems

The NP2G is designed for medium- to low-end systems, or "desktop routers," with an uplink to a larger switch. For high-end systems, the IBM PowerNP NP4GS3 Network Processor is required.

Systems developed with the NP2G use a distributed software model, which relies on a control point to execute software instructions. The control point can be an external microprocessor connected through an Ethernet link or the PCI interface. The NP2G's embedded PowerPC processor can also perform control point functions.

In this model, functions are divided between the control point and the network processor, as illustrated in *Figure 1-1*. The control point supports Layer 2 and Layer 3 routing protocols, Layer 4 and Layer 5 network applications, box maintenance, Management Information Base (MIB) collection (that is, the control point functions as an SNMP agent), and other systems management functions. Other functions, such as forwarding, filtering, and classification of the tables generated by the routing protocols, are performed by the dyadic protocol processor units (DPPUs) in each network processor in the system. The Core Language Processors (CLPs) in each DPPU execute the EPC's core software instruction set, which includes conditional execution, conditional branching, signed and unsigned operations, counts of leading zeros, and more.

Figure 1-1. Function Placement in an NP2G-Based System

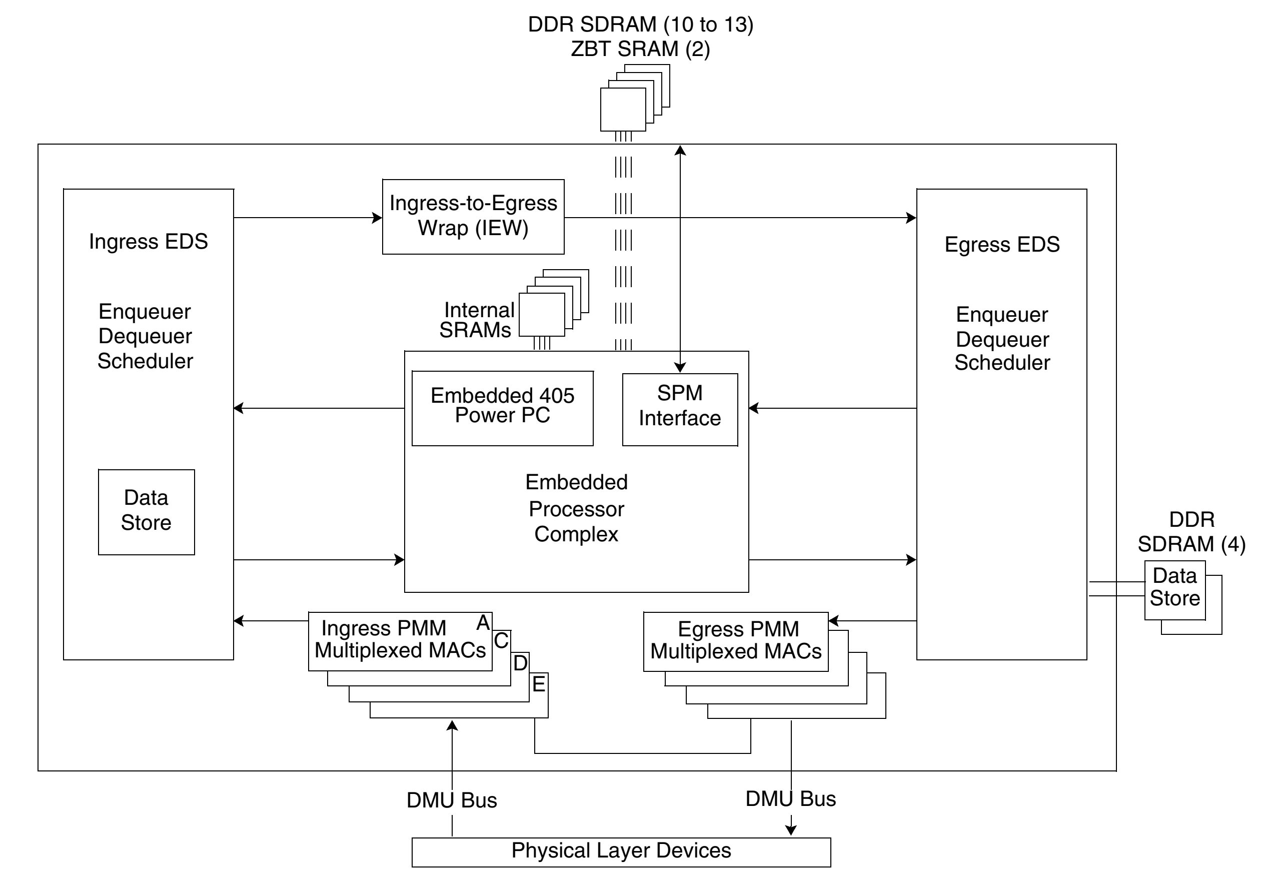


1.5 Structure

The IBM PowerNP NP2G network processor has seven major functional blocks, as illustrated in *Figure 1-2* on page 32:

EPC	Provides all processing functions for the device.
Embedded PowerPC™ Subsystem	Can act as a control point for the device; the Control Store interface provides up to 128 MB of program space for the embedded IBM PowerPC subsystem.
Ingress Enqueuer / Dequeuer / Scheduler (Ingress EDS)	Provides logic for frames traveling in from the physical layer devices.
Egress Enqueuer / Dequeuer / Scheduler (Egress EDS)	Provides logic for frames traveling out to the physical layer devices.
Ingress-to-Egress Wrap (IEW)	Transfers frames from the ingress side of the NP2G to the egress side of the NP2G.
Ingress Physical MAC Multiplexer (Ingress PMM)	Receives frames from physical layer devices.
Egress Physical MAC Multiplexer (Egress PMM)	Transmits frames to physical layer devices.

Figure 1-2. NP2G Major Functional Blocks



1.5.1 EPC Structure

The EPC contains six dyadic protocol processor units (DPPUs). Each DPPU contains two Core Language Processors (CLPs) that share 10 coprocessors, one coprocessor command bus, and a memory pool. The six DPPUs share 24 threads, four of which are enhanced, and three hardware accelerators.

Together, the six DPPUs are capable of operating on up to 24 frames in parallel. They share 32 K words of internal picocode instruction store, providing 1596 million instructions per second (MIPS) of processing power. In addition, the EPC contains a Hardware Classifier to parse frames on the fly, preparing them for processing by the picocode.

1.5.1.1 Coprocessors

Each DPPU contains two picocode processors, the CLPs, that execute the EPC's core instruction set and control thread swapping and instruction fetching. The two CLPs share eight dedicated coprocessors that can run in parallel with the CLPs:

Checksum
Calculates and verifies frame header checksums.

CAB Interface
Controls thread access to the Control Access Bus (CAB) through the CAB Arbiter; the CAB Control, CAB Arbiter, and CAB Interface enable debug access to NP2G data structures.

Coprocessor Response Bus
The coprocessor response bus (CRB) is a bus interface and an internal coprocessor that enables the attachment of an external coprocessor with results returned to an internal register. Picocode determines the processing status (busy/not busy) via the busy bit managed by the CRB coprocessor.

Counter
Updates counters for the picocode engines.

Data Store

- Interfaces frame buffer memory (ingress and egress directions), providing a 320-byte working area.
- Provides access to the Ingress and Egress Data Stores.

Enqueue
Manages control blocks containing key frame parameters; works with the Completion Unit hardware accelerator to enqueue frames to the IEW and target port output queues.

Policy
Determines if the incoming data stream complies with configured profiles.

String Copy
Accelerates data movement between coprocessors within the shared memory pool.

Tree Search Engine
Performs pattern analysis through tree searches (based on algorithms provided by the picocode) and read and write accesses, all protected by memory range checking; accesses Control Store memory independently.

Semaphore Manager
Assists in controlling access to shared resources, such as tables and control structures, through the use of semaphores; grants semaphores either in dispatch order (ordered semaphores) or in request order (unordered semaphores).

1.5.1.2 Enhanced Threads

Each CLP can run two threads, making four threads per DPPU, or 24 total. Twenty of the threads are General Data Handlers (GDHs), used for forwarding frames, and four of the 24 threads are enhanced:

Guided Frame Handler (GFH)
Handles Guided Frames, the in-band control mechanism between the EPC and all devices in the system, including the control point.

General Table Handler (GTH)
Builds table data in Control Memory

General PowerPC Handler Request (GPH-Req)	Processes frames bound to the embedded PowerPC.
General PowerPC Handler Response (GPH-Resp)	Processes responses from the embedded PowerPC.

1.5.1.3 Hardware Accelerators

The DPPUs share three hardware accelerators:

Completion Unit	Assures frame order as data exits the threads.
Dispatch Unit	Fetches data and parses the work out among the DPPUs.
Control Store Arbiter	Enables the processors to share access to the Control Store.

1.5.2 NP2G Memory

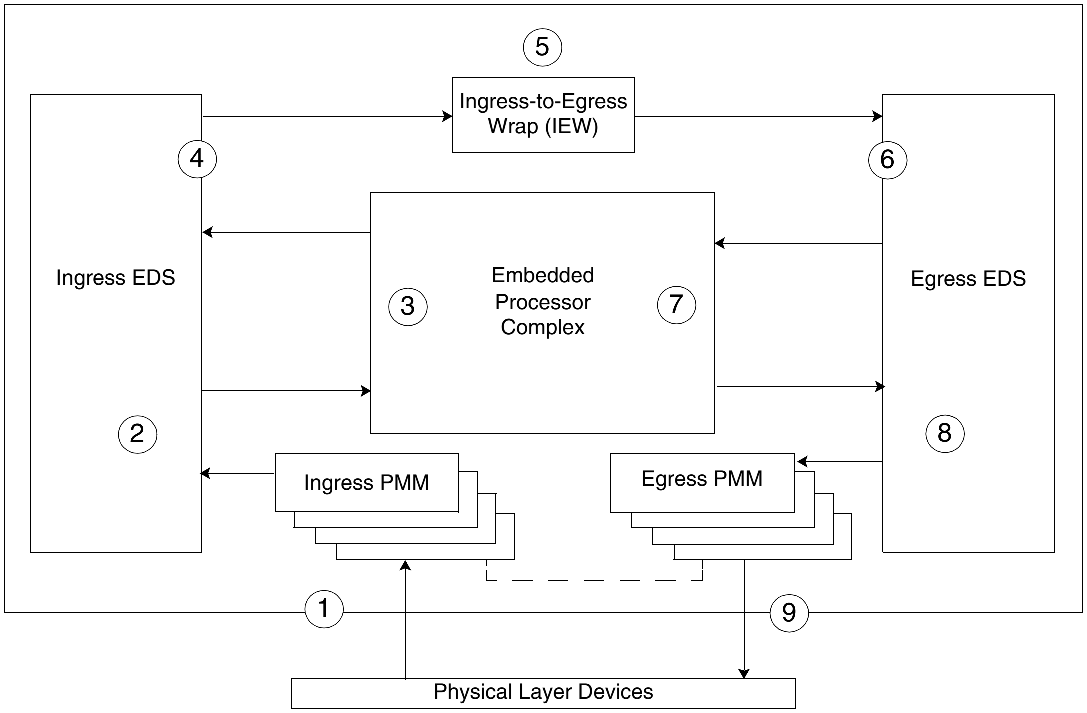
Storage for the NP2G is provided by both internal and external memories (see *Figure 1-2* on page 32). The Control Store contains all tables, counters, and any other data needed by the picocode. The Data Stores contain the frame data to be forwarded and can be used by the picocode (via the Data Store coprocessor) to create guided traffic.

The NP2G has the following stores:

- A common instruction memory that holds 32 K instruction words for normal processing and control functions
- 128 KB internal SRAM for input frame buffering
- 113 KB internal SRAM Control Store
- High capacity external DDR SDRAM for egress frame buffering and large forwarding tables; the amount of memory can vary depending on the configuration.
- External ZBT SRAM for fast table access
  - Up to 512 KB × 36 in the Z0 interface
  - Up to 123 KB × 18 in the Z1 interface (for use by the Scheduler)

1.6 Data Flow

Figure 1-3. Data Flow Overview



1.6.1 Basic Data Flow

Too many data flow routes and possibilities exist to fully document in this overview. However, data generally moves through the NP2G in the following manner (see *Figure 1-3*):

1. The Ingress PMM receives a frame from a physical layer device and forwards it to the Ingress EDS.
2. The Ingress EDS identifies the frame and enqueues it to the EPC.
3. The EPC processes the frame data (see *Section 1.6.2*).

   The EPC may discard the frame or modify the frame data directly and then return the updated data to the Ingress EDS's Data Store.
4. The frame is enqueued to the Ingress EDS, and the Ingress EDS Scheduler selects the frame for transmission and moves the data to the IEW.
5. The IEW transfers frames from the ingress EDS to the egress EDS.
6. The Egress EDS reassembles the frame and enqueues it to the EPC once it is fully reassembled.
7. The EPC processes it (see *Section 1.6.2*).

   The EPC may discard the frame or modify it using one of the frame alteration hardware assists. If extensive modification is required, the EPC may append to or rewrite the frame in Data Store.

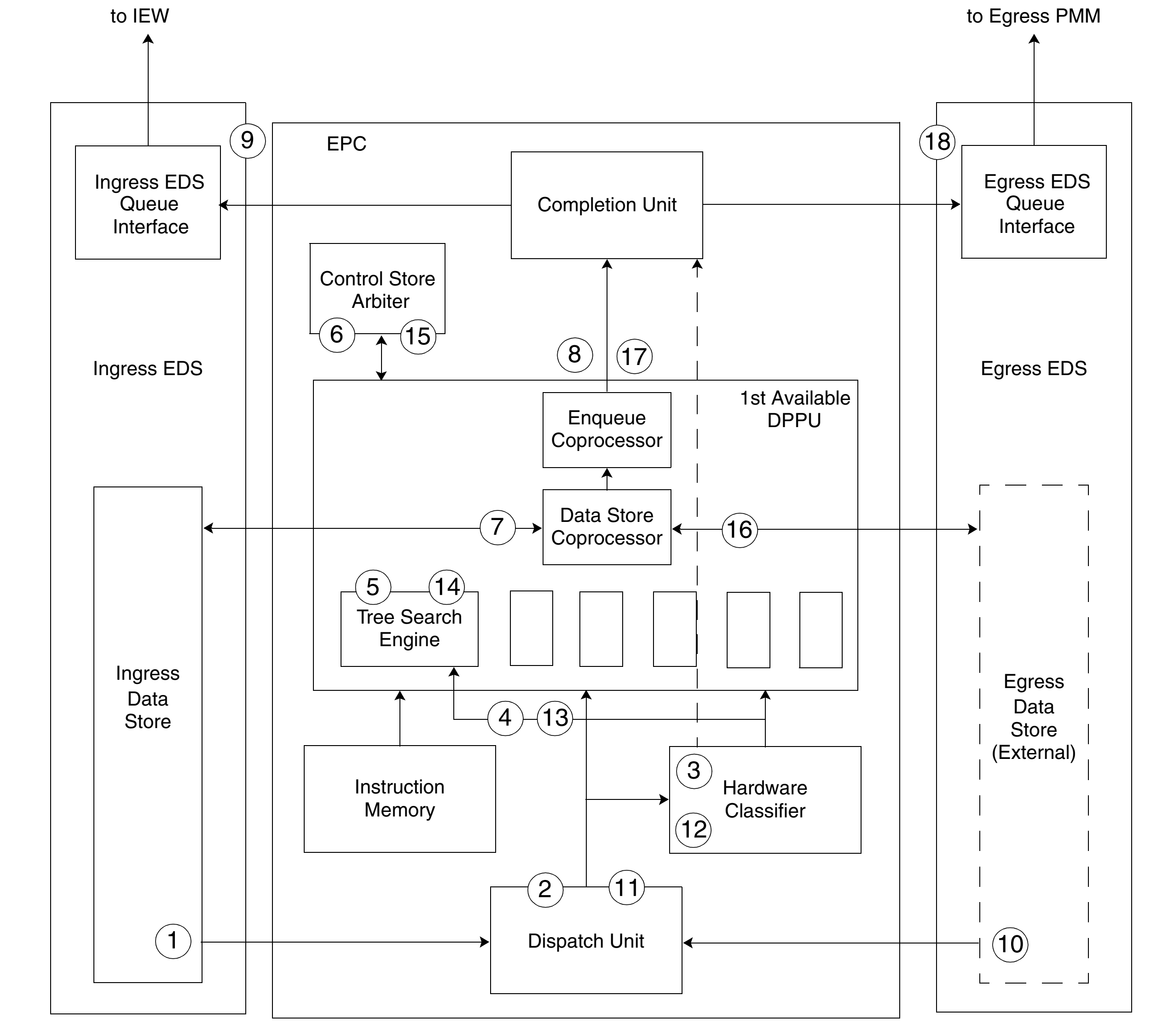
8. The frame is enqueued to the Egress EDS, and the Egress EDS Scheduler, if enabled, selects the frame for transmission and moves the data to the Egress PMM.

   If the Scheduler is not enabled, the EPC may forward the frame to a target port queue, to a wrap port, or to the GFH or GPH.

9. The Egress PMM sends the frame to a physical layer device.

1.6.2 Data Flow in the EPC

Figure 1-4. Basic Data Flow in the EPC



The EPC is the functional center of the device, and it plays a pivotal role in data flow. This section presents a basic overview of data flow in the EPC.

Ingress Side

1. The Ingress EDS enqueues a data frame to the EPC.
2. The Dispatch Unit fetches a portion of a frame and sends it to the next available thread.
3. Simultaneously, the Hardware Classifier (HC) determines the starting Common Instruction Address (CIA), parses different frame formats (for example: bridged, IP, and IPX), and forwards the results on to the thread.
4. The picocode examines the information from the HC and may examine the data further; it assembles search keys and launches the Tree Search Engine (TSE).
5. The TSE performs table searches, using search algorithms based on the format of the downloaded tables.
6. The Control Store Arbiter allocates Control Store memory bandwidth among the protocol processors.
7. Frame data moves into the Data Store coprocessor's memory buffer.
    - Forwarding and frame alteration information is identified by the results of the search.
    - The Ingress EDS can insert or overlay VLAN tags on the frame (hardware-assisted frame alteration) or the picocode can allocate or remove buffers to allow alteration of the frame (flexible frame alteration).
8. The Enqueue coprocessor builds the necessary information to enqueue the frame to the IEW and provides it to the Completion Unit (CU), which guarantees the frame order as the data moves from the 24 threads of the DPPU to the Ingress EDS queues.
9. The frame is enqueued to the Ingress EDS.
    - The Ingress EDS forwards the frame to the Ingress Scheduler.
    - The Scheduler selects the frame for transmission to the IEW.
      Note: The entire frame is not sent at once. The Scheduler sends it a cell at a time.
    - With the help of the Ingress EDS, the IEW segments the frames into 64-byte cells and inserts Cell Header and Frame Header bytes so that they can be stored and reassembled on the egress side. Thus, it is not necessary to use the smaller ingress data store to store frames while waiting for their ends to arrive.

Egress Side

10. The Egress EDS enqueues a data frame to the EPC.
11. The Dispatch Unit fetches a portion of a frame and sends it to the next available thread.
12. Simultaneously, the HC determines the starting CIA, parses different frame formats (for example: bridged, IP, and IPX), and forwards the results to the thread.
13. The picocode examines the information from the HC and may examine the data further; it assembles search keys and launches the TSE.
14. The TSE performs table searches, using search algorithms based on the format of the downloaded tables.
15. The Control Store Arbiter allocates Control Store memory bandwidth among the protocol processors.
16. Frame information, including alteration instructions, moves into queues in the Egress EDS; if flexible frame alteration is used, the Data Store coprocessor moves additional frame data to the Data Store.
    - Forwarding and frame alteration information is identified by the results of the search.
    - The NP2G provides two frame alteration techniques: hardware-assisted frame alteration and flexible frame alteration:

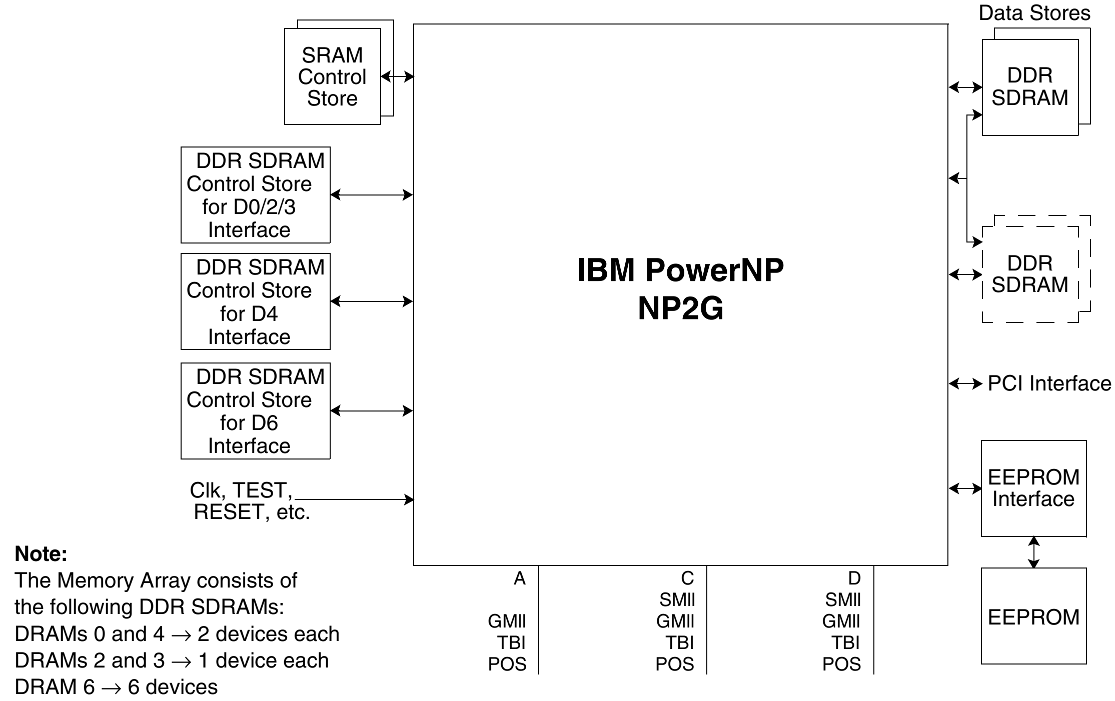
In hardware-assisted frame alteration, commands are passed to the Egress EDS hardware during enqueueing. These commands can, for example, update the TTL field in an IP header, generate frame CRC, or overlay an existing Layer 2 wrapper with a new one.

In flexible frame alteration, the picocode allocates additional buffers and the Data Store coprocessor places data into these buffers. The additional buffers allow prepending of data to a received frame and bypassing part of the received data when transmitting. This is useful for frame fragmentation when a when an IP header and MAC header must be prepended to received data in order to form a frame fragment of the correct size.

17. The Enqueue coprocessor builds the necessary information to enqueue the frame to the Egress EDS and provides it to the CU, which guarantees the frame order as the data moves from the 24 threads of the DPPU to the Egress EDS queues.
18. The frame is enqueued to the Egress EDS.
    - The frame is enqueued to the Egress EDS, which forwards it to the Egress Scheduler (if enabled).
    - The Scheduler selects the frame for transmission to a target port queue.
    - If the Scheduler is not enabled, the EDS will forward the frame directly to a target queue.
    - The Egress EDS selects frames for transmission from the target port queue and moves their data to the Egress PMM.

2. Physical Description

Figure 2-1. Device Interfaces



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

2.1 Pin Information

This section describes the many interfaces and associated pins of the NP2G Network Processor. For a summary of all the device's interfaces and how many pins each contains, see *Table 2-1*.

For information on signal pin locations, see *Table 2-38: Complete Signal Pin Listing by Signal Name* on page 88 and *Table 2-39: Complete Signal Pin Listing by Grid Position* on page 98.

The following table groups the interfaces and pins by function, briefly describes them, and points to the location of specific information in the chapter.

Table 2-1. Signal Pin Functions (Page 1 of 2)

Pin Type	Function	Resources
Flow Control		*Table 2-3: Flow Control Pins* on page 41
Z0 and Z1 ZBT SRAM Interface	Interface with the Z0 and Z1 ZBT SRAM for lookups	*Table 2-4: Z0 ZBT SRAM Interface Pins* on page 42 *Table 2-5: Z1 ZBT SRAM Interface Pins* on page 42 *Figure 2-2: ZBT SRAM Timing Diagram* on page 43
D3, D2, and D0 Memory	Interface with the DDR SDRAM used to implement the D3, D2, and D0 memories	*Table 2-10: D3 and D2 Interface Pins* on page 51 *Table 2-11: D0 Memory Pins* on page 52 *Figure 2-3: DDR Control Timing Diagram* on page 45 *Figure 2-4: DDR Read Timing Diagram* on page 46 *Figure 2-5: DDR Write Output Timing Diagram* on page 47
D4_0 and D4_1 Memory	Interface with the DDR DRAM used to implement the D4 memories	*Table 2-12: D4_0 and D4_1 Interface Pins* on page 53 *Figure 2-3: DDR Control Timing Diagram* on page 45 *Figure 2-4: DDR Read Timing Diagram* on page 46 *Figure 2-5: DDR Write Output Timing Diagram* on page 47
D6_5, D6_4, D6_3, D6_2, D6_1, and D6_0 Memory	Interface with the DDR SDRAM used to implement the PowerPC Store	*Table 2-13: D6_5, D6_4, D6_3, D6_2, D6_1, and D6_0 Memory Pins* on page 54 *Figure 2-3: DDR Control Timing Diagram* on page 45 *Figure 2-4: DDR Read Timing Diagram* on page 46 *Figure 2-5: DDR Write Output Timing Diagram* on page 47
DS1 and DS0 Memory	Interface with the DDR DRAM used to implement the DS1 and DS0 memories	*Table 2-14: DS1 and DS0 Interface Pins* on page 55 *Figure 2-3: DDR Control Timing Diagram* on page 45 *Figure 2-4: DDR Read Timing Diagram* on page 46 *Figure 2-5: DDR Write Output Timing Diagram* on page 47
PMM Interface	Interface with the physical layer devices through the following buses:	*Table 2-15: PMM Interface Pins* on page 56 *Table 2-16: PMM Interface Pin Multiplexing* on page 57 *Figure 2-6: NP2G DMU Bus Clock Connections* on page 58
	TBI	*Table 2-18: Parallel Data Bit to 8B/10B Position Mapping (TBI Interface)* on page 59 *Table 2-19: PMM Interface Pins: TBI Mode* on page 59 *Figure 2-8: TBI Timing Diagram* on page 61
	GMII	*Table 2-21: PMM Interface Pins: GMII Mode* on page 63 *Figure 2-9: GMII Timing Diagram* on page 64
	SMII	*Table 2-23: PMM Interface Pins: SMII Mode* on page 65 *Figure 2-10: SMII Timing Diagram* on page 66
	POS	*Figure 2-7: NP2G DMU Bus Clock Connections (POS Overview)* on page 59 *Table 2-25: POS Signals* on page 67 *Figure 2-11: POS Transmit Timing Diagram* on page 68 *Figure 2-12: POS Receive Timing Diagram* on page 69

Table 2-1. Signal Pin Functions (Page 2 of 2)

Pin Type	Function	Resources
PCI Interface	Interface to the PCI bus	*Table 2-27: PCI Pins* on page 71 *Figure 2-13: PCI Timing Diagram* on page 72
Management Bus	Translated into various “host” buses by an external FPGA (SPM)	*Table 2-29: Management Bus Pins* on page 73 *Figure 2-14: SPM Bus Timing Diagram* on page 74
Miscellaneous	Various interfaces	*Table 2-31: Miscellaneous Pins* on page 75 *Table 2-32: Signals Requiring Pull-Up or Pull-Down* on page 77

2.1.1 Ingress-to-Egress Wrap (IEW) Pins

Table 2-2. Ingress-to-Egress Wrap (IEW) Pins

Signal (Clock Domain)	Description	Type

Master_Grant_A(1:0)
Master_Grant_B(1:0)
(IEW Clk * 2)
	These I/O signals are used to receive status information from the egress side. Both “A” bits should be connected to the Send_Grant_A pin and both “B” bits should be connected to the Send_Grant_B pin.	Input
		5.0 V-tolerant
		3.3 V LVTTL

Send_Grant_A Send_Grant_B (IEW Clk * 2)	Send Grants A and B indicate whether the IEW is able to receive cells from the ingress side. 0 Unable 1 Able The NP2G changes the state of these signals for use on the corresponding Master_Grant inputs.	Output 5.0 V-tolerant 3.3 V LVTTL

2.1.2 Flow Control Interface Pins

Table 2-3. 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
RES_Sync	Remote Egress Status synchronization (sync) may be driven by the network processor to indicate the start of time division multiplex cycle for passing egress status information to the ingress side.	Input/Output 5.0 V-tolerant 3.3 V LVTTL
RES_Data	Remote Egress Status data is driven by the network processor to indicate its congestion status to the ingress side during a 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

2.1.3 ZBT Interface Pins

These pins interface with Z0 and Z1 ZBT SRAM for lookups as described in *Table 2-4* and *Table 2-5*.

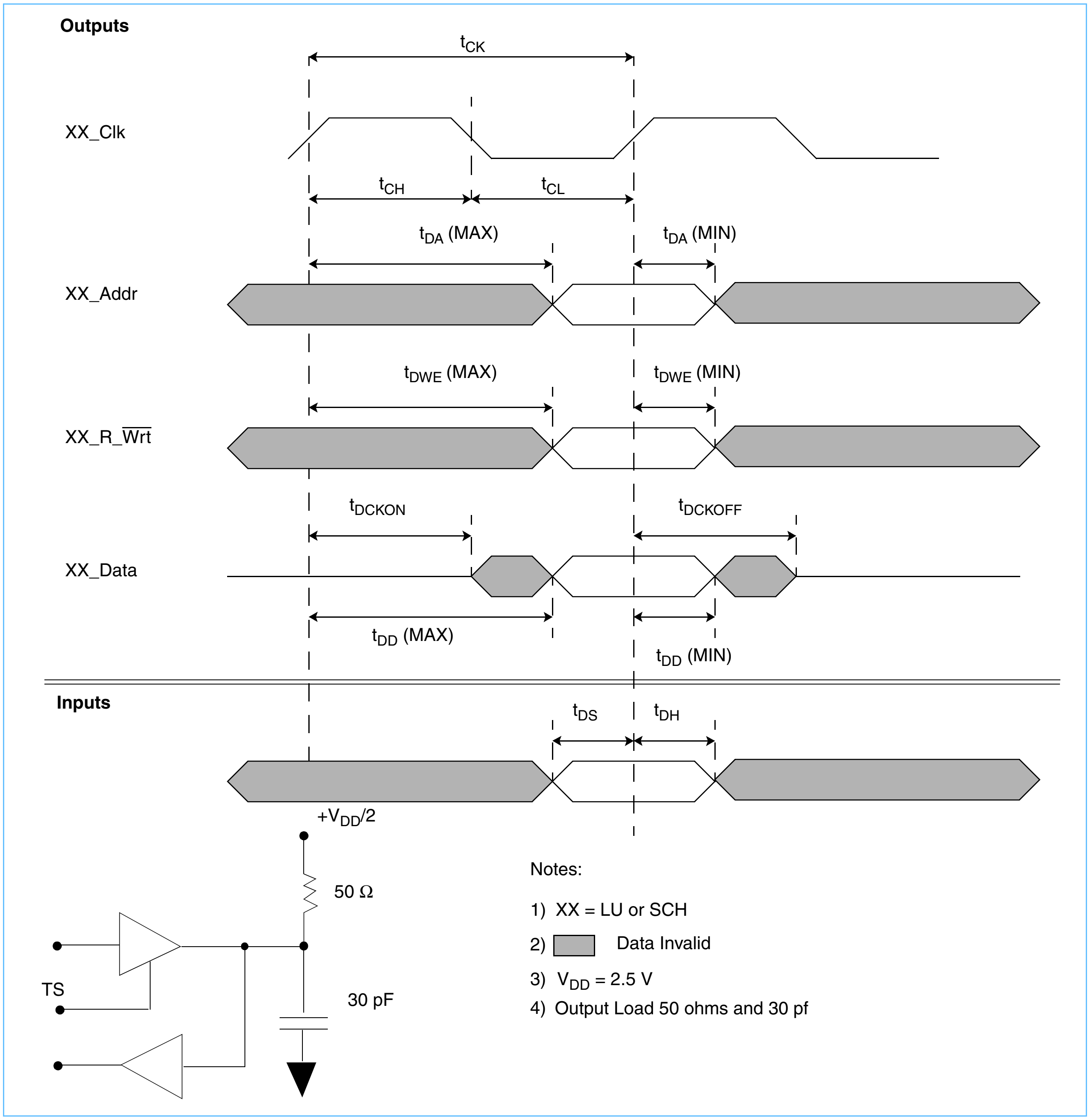
Table 2-4. Z0 ZBT SRAM Interface Pins

Signal	Description	Type
LU_Clk	Look-Up clock. 7.5 ns period (133 MHz).	Output CMOS 2.5 V
LU_Addr(19:0)	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
cam_cp_response(13:0)	Coprocessor Response Bus (CRB). Results from an external coprocessor are sent to a specified thread's CRB Coprocessor and stored into the CRB Results register.	Input/Output CMOS 2.5 V

Table 2-5. 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)	Data bus. When used as SRAM input, the rising edge of SCH_Clk samples these signals.	Input/Output CMOS 2.5 V
SCH_R_$\overline{\text{Wrt}}$	Read/Write control signal is sampled by the rising edge of SCH_Clk. 0 Write 1 Read	Output CMOS 2.5 V

Figure 2-2. ZBT SRAM Timing Diagram



Notes:

1) XX = LU or SCH

2) ▨ Data Invalid

3) V_{DD} = 2.5 V

4) Output Load 50 ohms and 30 pf

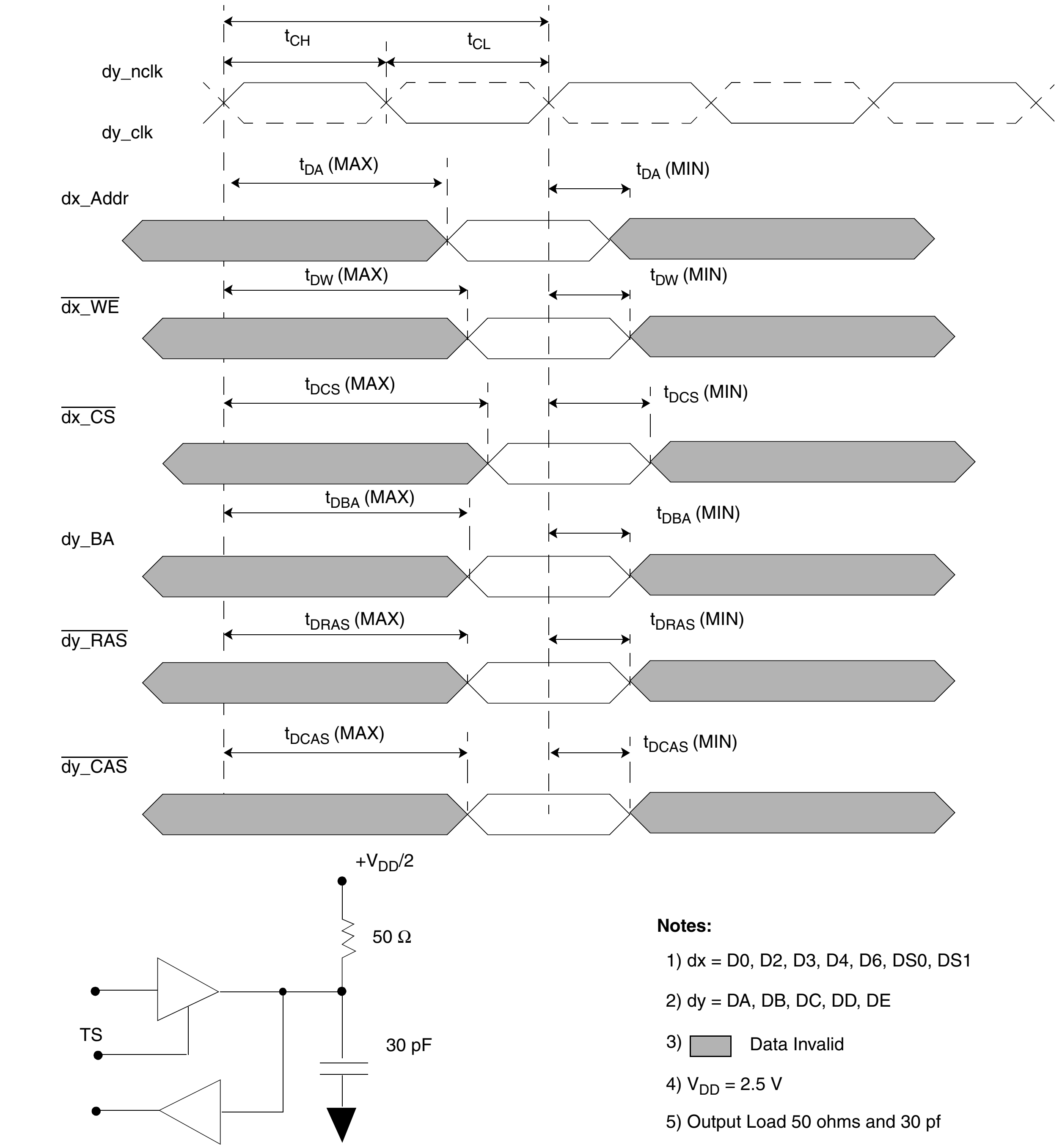
Table 2-6. ZBT SRAM Timing Diagram Legend (for Figure 2-2)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	ZBT Cycle Time	7.5	
t_{CH}	Clock Pulse Width High	3.5	3.7
t_{CL}	Clock Pulse Width Low	3.8	4.0
t_{DA}	Address Output Delay	1.0	3.0
t_{DWE}	Read/Write Output Delay	1.1	2.6
t_{DD}	Data Output Delay	1.1	2.7
t_{DCKON}	Data Output Turn On	1.3	4.1
t_{DCKOFF}	Data Output Turn Off	0.8	3.0
t_{DS}	Input Data Setup Time	1.0	
t_{DH}	Input Data Hold Time	0	
Note: All delays are measured with 1 ns slew time measured from 10 - 90% of input voltage.			

2.1.4 DDR DRAM Interface Pins

The pins described here interface with DDR DRAM to implement data store, control store, and the PowerPC store. The control, read, and write timing diagrams (*Figure 2-3*, *Figure 2-4*, and *Figure 2-5*) apply to all pin tables in this section.

Figure 2-3. DDR Control Timing Diagram



Notes:

1) dx = D0, D2, D3, D4, D6, DS0, DS1

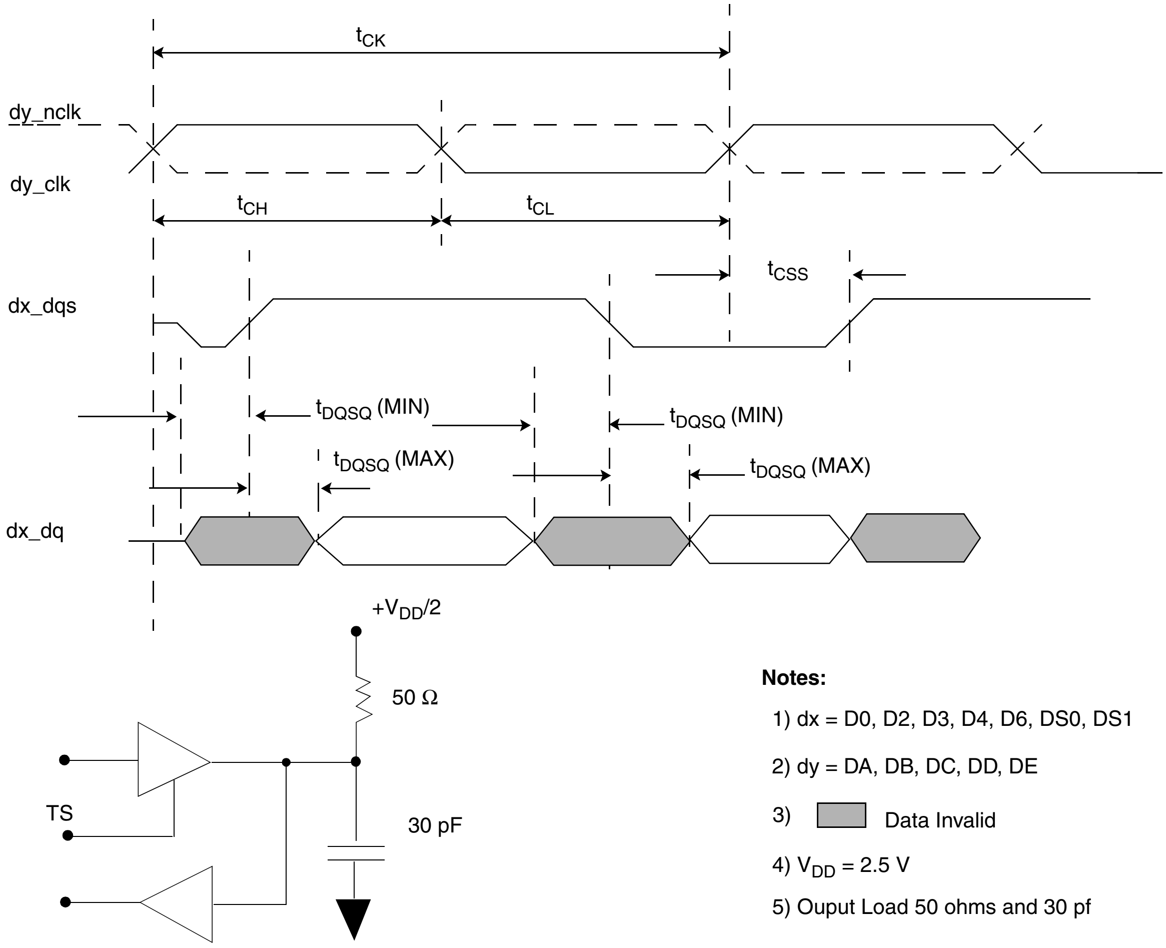
2) dy = DA, DB, DC, DD, DE

3) ▨ Data Invalid

4) V_{DD} = 2.5 V

5) Output Load 50 ohms and 30 pf

Figure 2-4. DDR Read Timing Diagram



Notes:

1) dx = D0, D2, D3, D4, D6, DS0, DS1

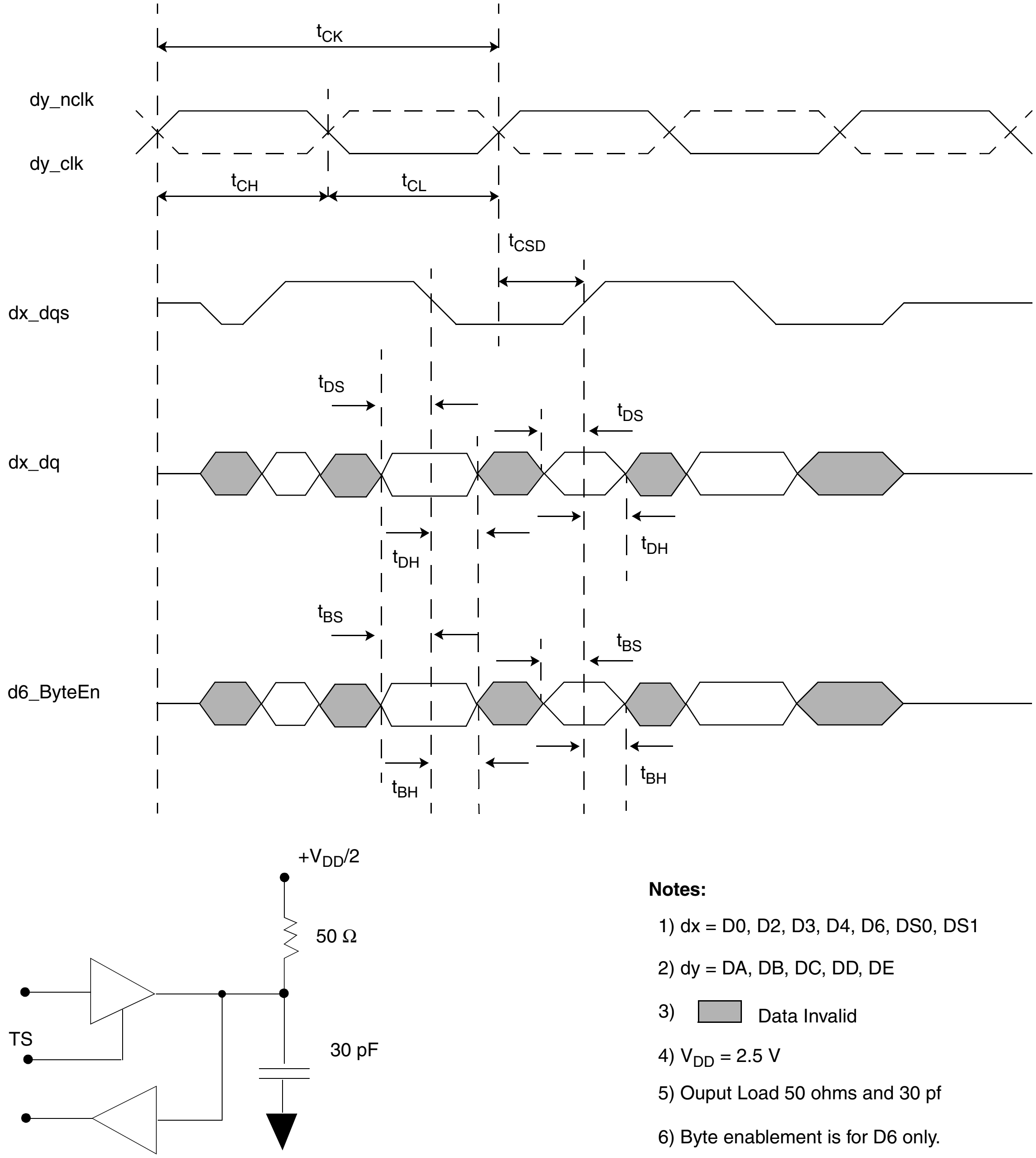
2) dy = DA, DB, DC, DD, DE

3) ▨ Data Invalid

4) V_{DD} = 2.5 V

5) Ouput Load 50 ohms and 30 pf

Figure 2-5. DDR Write Output Timing Diagram



Notes:

1) dx = D0, D2, D3, D4, D6, DS0, DS1

2) dy = DA, DB, DC, DD, DE

3) ▨ Data Invalid

4) V_{DD} = 2.5 V

5) Ouput Load 50 ohms and 30 pf

6) Byte enablement is for D6 only.

Table 2-7. DDR Timing Diagram Legend (for Figure 2-3, Figure 2-4, and Figure 2-5) Values are for D0, D2, D3, D4, DS0, and DS1

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	DDR Clock Cycle Time	7.5	
t_{CH}	Clock Pulse Width High	0.45 * t_{CK}	0.55 * t_{CK}
t_{CL}	Clock Pulse Width Low	0.45 * t_{CK}	0.55 * t_{CK}
t_{DA}	Address Output Delay	1.7	4.8
t_{DW}	Write Enable Output Delay	2.0	5.2
t_{DCS}	Chip Select Output Delay	1.9	5.2
t_{BA}	Bank Address Output Delay	1.9	5.2
t_{DRAS}	RAS Output Delay	1.9	5.4
t_{DCAS}	CAS Output Delay	1.9	5.4
t_{CSD}	dy_clk to dx_dqs Strobe Output Delay	0.5	1.4
t_{DS}	Data to Strobe Output Setup Time	1.0	
t_{DH}	Data to Strobe Output Hold Time	1.2	
t_{CSS}	Clock to Strobe Input Skew	-2.0	2.5
t_{DQSQ}	Data to DQS Input Skew	1.0	1.1

Note: All delays are measured with 1 ns skew time measured from 10-90% of input voltage.
All measurements made with Test Load of 50 ohms and 30 pf.
The dx_dqs pin descriptions for the DS0, DS1, and D4 interfaces (*Section 2.1.4.2* on page 53 and *Section 2.1.4.4* on page 55) describe the association of data strobe to data. This association is dependent on the setting of Strobe_cntl in the DRAM configuration register is described in *Section 13.1.2* on page 434.

Table 2-8. DDR Timing Diagram Legend (for Figure 2-3, Figure 2-4, and Figure 2-5) Values are for D6 in 4-bit Interface Mode.

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	DDR Clock Cycle Time	7.5	
t_{CH}	Clock Pulse Width High	0.45 * t_{CK}	0.55 * t_{CK}
t_{CL}	Clock Pulse Width Low	0.45 * t_{CK}	0.55 * t_{CK}
t_{DA}	Address Output Delay	2.4	6.0
t_{DW}	Write Enable Output Delay	2.4	5.6
t_{DCS}	Chip Select Output Delay	2.5	5.9
t_{BA}	Bank Address Output Delay	2.3	5.7
t_{DRAS}	RAS Output Delay	2.4	5.7
t_{DCAS}	CAS Output Delay	2.5	5.7
t_{CSD}	dy_clk to dx_dqs Strobe Output Delay	0.7	1.6
t_{DS}	Data to Strobe Output Setup Time	1.5	
t_{DH}	Data to Strobe Output Hold Time	1.0	
t_{BS}	Byte Enable to Strobe Setup Time	1.1	
t_{BH}	Byte Enable to Strobe Hold Time	1.3	
t_{CSS}	Clock to Strobe Input Skew	-2.0	3.0
t_{DQSQ}	Data to DQS Input Skew for d6_data_00-03	1.0	1.3
	Data to DQS Input Skew for d6_data_04-07	0.9	1.3
	Data to DQS Input Skew for d6_data_08-11	1.0	1.4
	Data to DQS Input Skew for d6_data_12-15	1.0	1.3
	Data to DQS Input Skew for d6_parity_00	1.7	1.1
	Data to DQS Input Skew for d6_parity_01	1.7	1.1

Note: All delays are measured with 1 ns slew time measured from 10-90% of input voltage.
All measurements made with Test Load of 50 ohms and 30 pf.

Table 2-9. DDR Timing Diagram Legend (for Figure 2-3, Figure 2-4, and Figure 2-5) Values are for D6 in 16-bit Interface Mode.

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	DDR Clock Cycle Time	7.5	
t_{CH}	Clock Pulse Width High	0.45 * t_{CK}	0.55 * t_{CK}
t_{CL}	Clock Pulse Width Low	0.45 * t_{CK}	0.55 * t_{CK}
t_{DA}	Address Output Delay	2.4	6.0
t_{DW}	Write Enable Output Delay	2.4	5.6
t_{DCS}	Chip Select Output Delay	2.5	5.9
t_{BA}	Bank Address Output Delay	2.3	5.7
t_{DRAS}	RAS Output Delay	2.4	5.7
t_{DCAS}	CAS Output Delay	2.5	5.7
t_{CSD}	dy_clk to dx_dqs Strobe Output Delay	0.8	1.6
t_{DS}	Data to Strobe Output Setup Time	1.5	
t_{DH}	Data to Strobe Output Hold Time	1.0	
t_{BS}	Byte Enable to Strobe Setup Time	1.1	
t_{BH}	Byte Enable to Strobe Hold Time	1.3	
t_{CSS}	Clock to Strobe Input Skew	-2.1	2.9
t_{DQSQ}	Data to DQS Input Skew for d6_data_00-07	0.8	1.3
	Data to DQS Input Skew for d6_data_08-15	0.9	1.4
	Data to DQS Input Skew for d6_parity_00	1.7	1.1
	Data to DQS Input Skew for d6_parity_01	1.7	1.1

Note: All delays are measured with 1 ns slew time measured from 10-90% of input voltage.
All measurements made with Test Load of 50 ohms and 30 pf.

2.1.4.1 D3, D2, and D0 Interface Pins

These pins interface with the DDR SDRAM used to implement the D3, D2, and D0 control stores.

Table 2-10. D3 and D2 Interface Pins

Signal	Description	Type
Shared Signals		
DB_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D3 and D2 memory devices.	Output SSTL2 2.5 V
$\overline{\mathrm{DB\_Clk}}$	The negative pin of an output differential pair. 133 MHz. Common to the D3 and D2 memory devices.	Output SSTL2 2.5 V
$\overline{\mathrm{DB\_RAS}}$	Common row address strobe (common to D3 and D2).	Output SSTL2 2.5 V
$\overline{\mathrm{DB\_CAS}}$	Common column address strobe (common to D3 and D2).	Output SSTL2 2.5 V
DB_BA(1:0)	Common bank address (common to D3 and D2).	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
$\overline{\mathrm{D3\_WE}}$	D3 write enable	Output CMOS 2.5 V
$\overline{\mathrm{D3\_CS}}$	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
$\overline{\mathrm{D2\_WE}}$	D2 write enable	Output CMOS 2.5 V
$\overline{\mathrm{D2\_CS}}$	D2 chip select	Output CMOS 2.5 V

Table 2-11. 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
$\overline{\text{DE\_RAS}}$	Common row address strobe	Output CMOS 2.5 V
$\overline{\text{DE\_CAS}}$	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
$\overline{\text{D0\_WE}}$	D0 write enable	Output CMOS 2.5 V
$\overline{\text{D0\_CS}}$	D0 chip select	Output CMOS 2.5 V

2.1.4.2 D4_0 and D4_1 Interface Pins

These pins interface with the DDR DRAM used to implement the D4 control store.

Table 2-12. D4_0 and D4_1 Interface 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
$\overline{\text{DD\_RAS}}$	Common row address strobe	Output CMOS 2.5 V
$\overline{\text{DD\_CAS}}$	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. Data bits are associated with strobe bits as follows: Strobe_cntl = ‘01’ Strobe_cntl = ‘00’ 3 — 31:24 2 — 23:16 1 — 15:8 0 31:0 7:0	Input/Output SSTL2 2.5 V
D4_Data(31:0)	D4 data bus	Input/Output SSTL2 2.5 V
$\overline{\text{D4\_WE}}$	D4 write enable	Output CMOS 2.5 V
$\overline{\text{D4\_CS}}$	D4 chip select	Output CMOS 2.5 V

2.1.4.3 D6_*x* Interface Pins

These pins interface with the DDR SDRAM used to implement the PowerPC store.

Table 2-13. D6_5, D6_4, D6_3, D6_2, D6_1, and D6_0 Memory Pins

Signal	Description	Type
DA_Clk	The positive pin of an output differential pair. 133 MHz. Common to the D6 memory devices.	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
$\overline{\text{DA\_RAS}}$	Common row address strobe (common to D6).	Output SSTL2 2.5 V
$\overline{\text{DA\_CAS}}$	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
$\overline{\text{D6\_WE}}$	Common write enable (common to D6).	Output SSTL2 2.5 V
D6_Addr(12:0)	D6 address	Output SSTL2 2.5 V
$\overline{\text{D6\_CS}}$	D6 chip select	Output SSTL2 2.5 V
D6_DQS(3:0)	D6 data strobes. Data bits are associated with strobe bits as follows: D6_DRAM_Size = '0xx' D6_DRAM_Size = '1xx' 3 — 15:12 2 15:8 11:8 1 — 7:4 0 7:0 3:0	Input/Output SSTL2 2.5 V
D6_Data(15:0)	D6 data bus	Input/Output SSTL2 2.5 V
$\overline{\text{D6\_ByteEn}}$(1:0)	D6 byte enables byte masking write to D6. Data is masked when D6_ByteEn is high. Data bits are associated with byte enable as follows: 1 15:8 0 7:0	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. Data bits are associated with parity bits as follows: 1 15:8 0 7:0	Input/Output SSTL2 2.5 V
D6_DQS_Par(1:0)	D6 data strobe for the parity signals	Input/Output SSTL2 2.5 V

2.1.4.4 DS1 and DS0 Interface Pins

These pins interface with the DDR DRAM used to implement the DS1 and DSO data stores.

Table 2-14. DS1 and DS0 Interface Pins (Page 1 of 2)

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
$\overline{\text{DC\_RAS}}$	Common Row address strobe (common to DS1 and DS0).	Output SSTL2 2.5 V
$\overline{\text{DC\_CAS}}$	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. Data bits are associated with strobe bits as follows: Strobe_cntl = ‘01’ Strobe_cntl = ‘00’ 3 — 31:24 2 — 23:16 1 — 15:8 0 31:0 7:0	Input/Output SSTL2 2.5 V
DS1_Data(31:0)	DS1 data bus	Input/Output SSTL2 2.5 V
$\overline{\text{DS1\_WE}}$	DS1 write enable	Output CMOS 2.5 V
$\overline{\text{DS1\_CS}}$	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. Data bits are associated with strobe bits as follows: Strobe_cntl = ‘01’ Strobe_cntl = ‘00’ 3 — 31:24 2 — 23:16 1 — 15:8 0 31:0 7:0	Input/Output SSTL2 2.5 V

Table 2-14. DS1 and DS0 Interface Pins (Page 2 of 2)

Signal	Description	Type
DS0_Data(31:0)	DS0 data bus	Input/Output SSTL2 2.5 V
$\overline{\text{DS0\_WE}}$	DS0 write enable	Output CMOS 2.5 V
$\overline{\text{DS0\_CS}}$	DS0 chip select	Output CMOS 2.5 V

2.1.5 PMM Interface Pins

These pins allow the Physical MAC Multiplexer (PMM) to interface with the physical layer devices. The NP2G has different sets of pins for the Ten-Bit (TBI), Gigabit Media-Independent (GMII), Serial Media-Independent (SMII), and Packet over SONET (POS) interfaces.

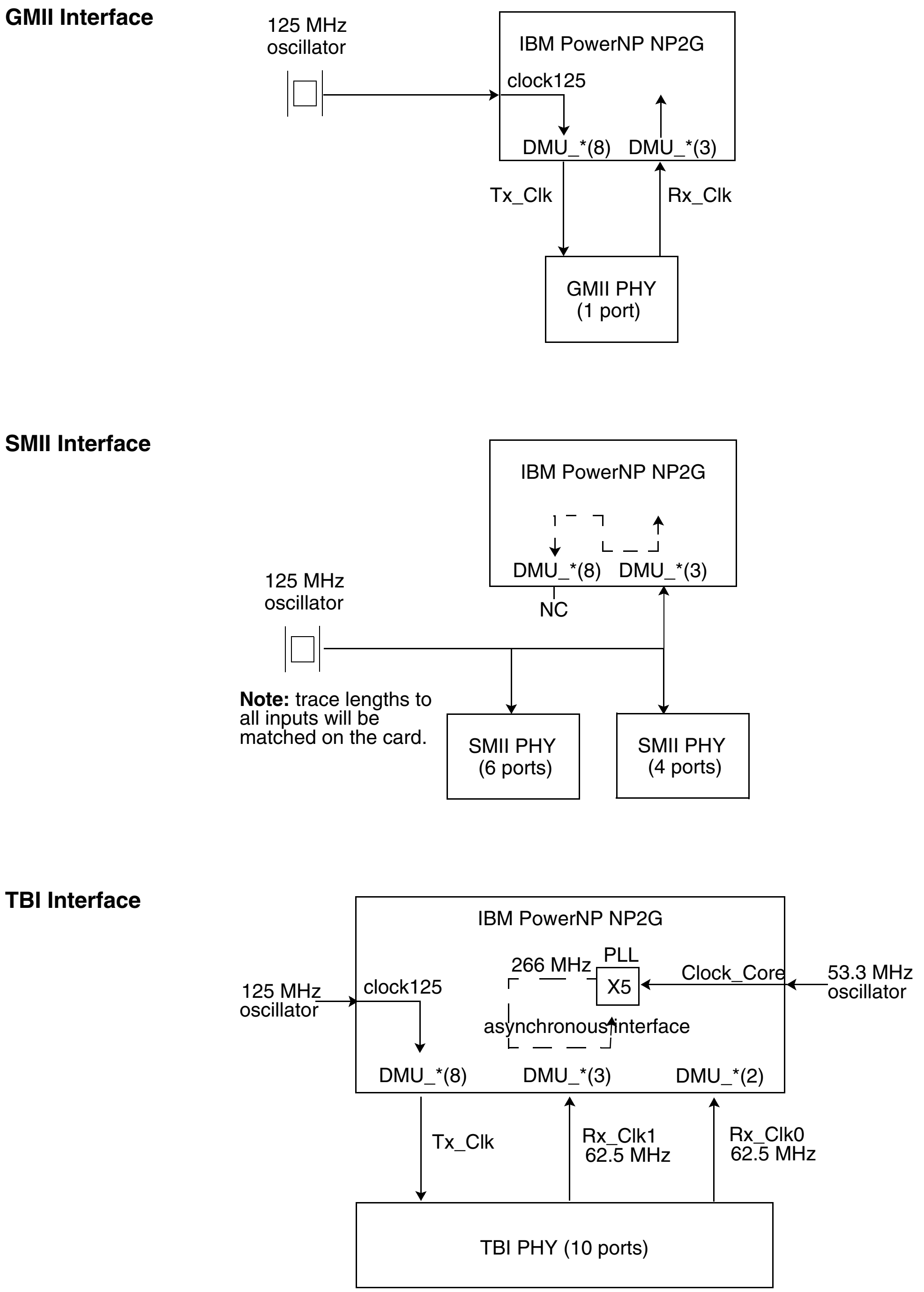
Table 2-15. PMM Interface Pins

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 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL
DMU_C(30:0)	Define the third of the four PMM interfaces and can be configured for TBI, SMII, GMII, or POS. See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL
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 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions and definitions.	5.0 V-tolerant 3.3 V LVTTL

Table 2-16. PMM Interface Pin Multiplexing

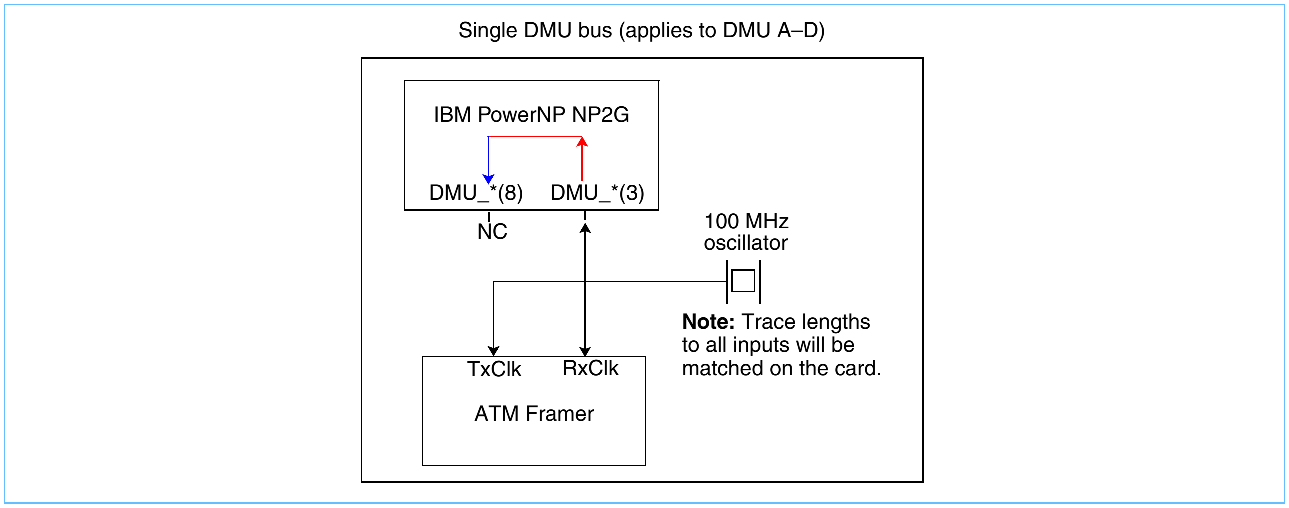
Pin(s)	Pin Mode		Interface Type				
	DMU_A, DMU_C	DMU_D	GMII	TBI	SMII	Debug (DMU_D only)	8-Bit POS
30	O	O					RxAddr(1) O
29	O	O					RxAddr(0) O
28	O	O					TxAddr(1) O
27	O	O					TxAddr(0) O
26	O	O					TxSOF O
25	O	O	Tx_Valid_Byte O				TxEOF O
(24:17)	O	O	Tx_Data(7:0) O	Tx_Data(0:7) O	Tx_Data(9:2) O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

Figure 2-6. NP2G DMU Bus Clock Connections



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

Figure 2-7. NP2G DMU Bus Clock Connections (POS Overview)


Single DMU bus (applies to DMU A–D)
IBM PowerNP NP2G
DMU_*(8)
DMU_*(3)
NC
100 MHz oscillator
Note: Trace lengths to all inputs will be matched on the card.
TxClk
RxClk
ATM Framer

2.1.5.1 TBI Bus Pins

Table 2-17. PMM Interface Pins: Debug (DMU_D Only)

Signal	Description	Type
Debug (23:0)	When DMU_D is configured as the debug bus, signals internal to the NP2G are available to be observed externally. This mode is supported only when its use is directed by an IBM Network Processor Application Engineer. It is recommended that board designs provide for the attachment of scope probes to observe this interface which runs at 133 MHz.	Output 5.0 V-tolerant 3.3 V LVTTL

Table 2-18. 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

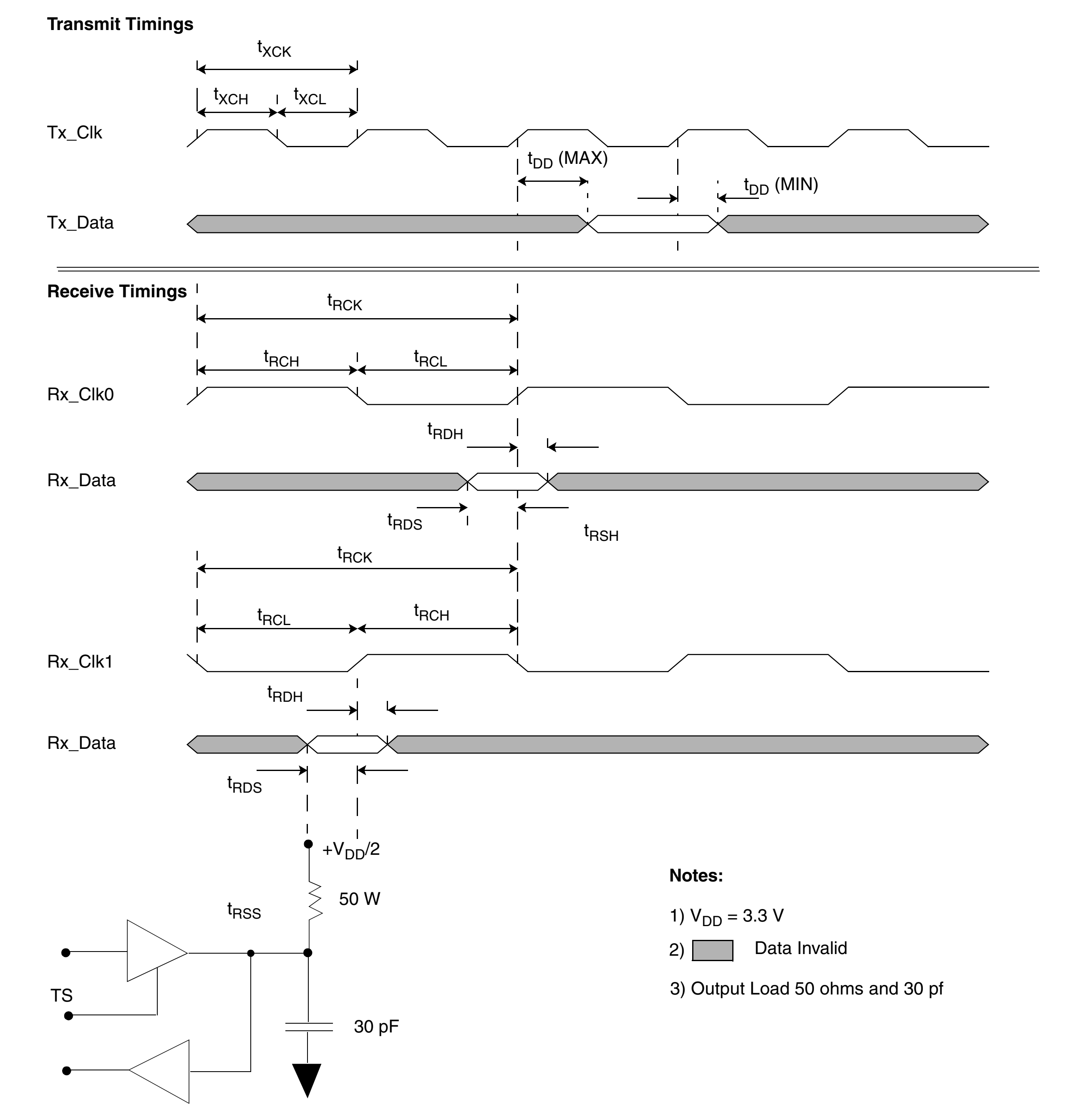
Table 2-19. PMM Interface Pins: TBI Mode (Page 1 of 2)

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
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
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
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
Note: See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions (I/O) and definitions.		

Table 2-19. PMM Interface Pins: TBI Mode (Page 2 of 2)

Signal	Description	Type
Sig_Det	Signal Detect. Signal asserted by the PHY to indicate that the physical media are valid.	Input 5.0 V-tolerant 3.3 V LVTTL
$\overline{\text{CPDetect}}$	This pin must be pulled low to indicate when a Control Point is active on this DMU interface. Otherwise, this signal should be pulled high (inactive). The state of the signal is captured in the Data Mover Unit (DMU) Configuration (page 485) when the $\overline{\text{Reset}}$ signal is deasserted. After DMU configuration, this signal is driven by the network processor to indicate the status of the TBI 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
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
Note: See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions (I/O) and definitions.		

Figure 2-8. TBI Timing Diagram



Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Table 2-20. TBI Timing Diagram Legend (for Figure 2-8)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{XCK}	Tx_Clk Transmit Cycle Time	8	
t_{XCH}	Tx_Clk Pulse Width High	3.5	4.0
t_{XCL}	Tx_Clk Pulse Width Low	4.0	4.5
t_{DD}	Tx_Data_(7:0) Output Delay	3.2	4.7
t_{RCK}	Rx_Clk0/Rx_Clk1 Receive Cycle Time	16	
t_{RCH}	Rx_Clk0/Rx_Clk1 Pulse Width High	7	
t_{RCL}	Rx_Clk0/Rx_Clk1 Pulse Width Low	7	
t_{RDS}	Rx_Data_(9:0) Setup Time Clk0	0.7	
t_{RDH}	Rx_Data_(9:0) Hold Time Clk0	1.2	
t_{RDS}	Rx_Data_(9:0) Setup Time Clk1	0.7	
t_{RDH}	Rx_Data_(9:0) Hold Time Clk1	1.2	

Note: All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

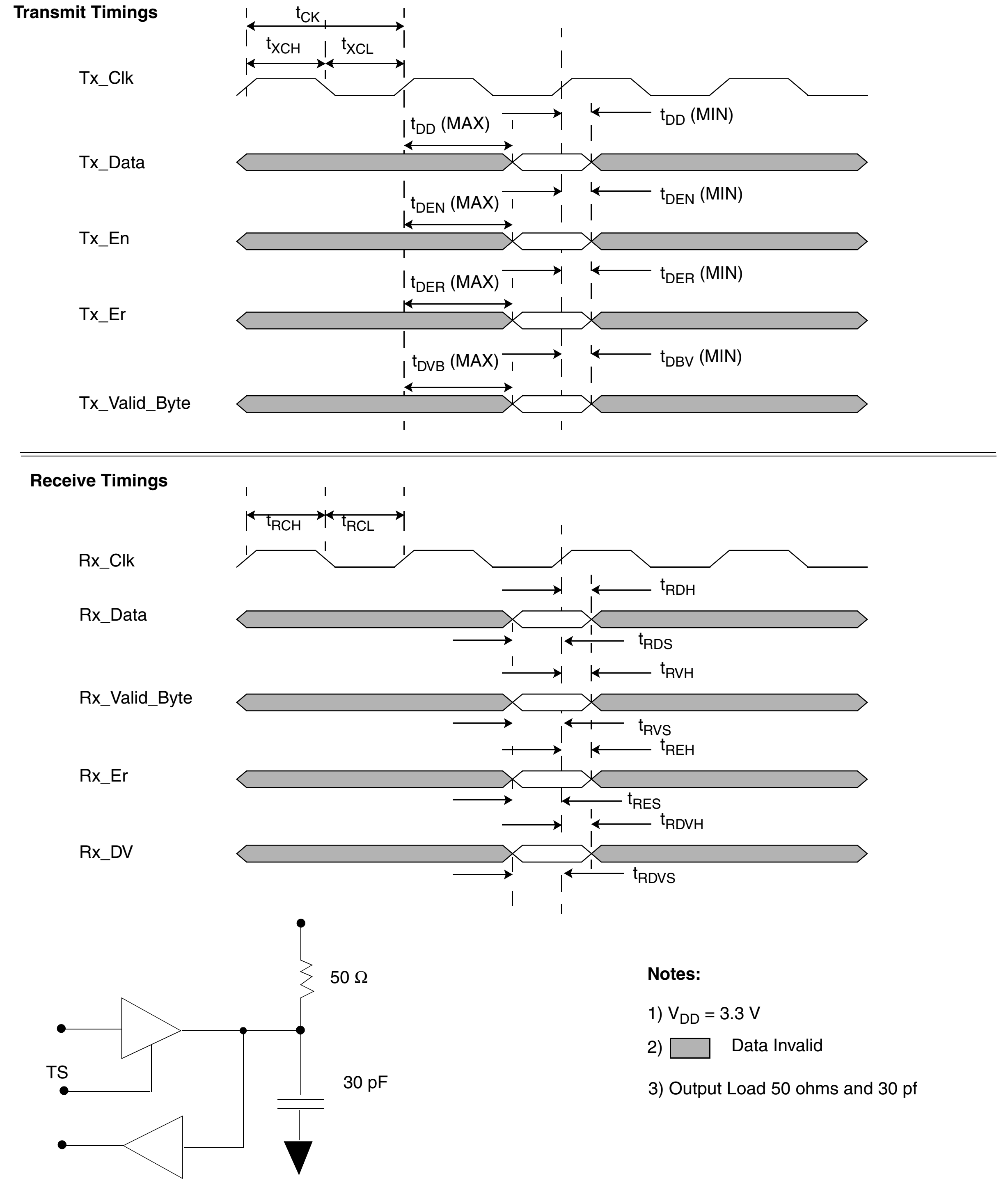
2.1.5.2 GMII Bus Pins

Table 2-21. PMM Interface Pins: GMII Mode

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
Rx_Data(7:0)	Received Data. Data bus from the PHY, synchronous to Rx_Clk.	Input 5.0 V-tolerant 3.3 V LVTTL
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
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
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
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
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
Rx_Clk	125 MHz Receive Medium clock generated by the PHY.	Input 5.0 V-tolerant 3.3 V LVTTL
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
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
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
$\overline{\text{CPDetect}}$	This pin must be pulled low to indicate when a Control Point is active on this DMU interface. Otherwise, this signal should be pulled high (inactive). The state $\overline{\text{of the}}$ signal is captured in the Data Mover Unit (DMU) Configuration (page 485) when the $\overline{\text{Reset}}$ signal is deasserted.	Input 5.0 V-tolerant 3.3 V LVTTL

Note: The NP2G supports GMII in Full-Duplex Mode only.
See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for pin directions (I/O) and definitions.

Figure 2-9. GMII Timing Diagram



Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Table 2-22. GMII Timing Diagram Legend (for Figure 2-9)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	Tx_Clk Cycle Time	8	
t_{XCH}	Transmit Clock Pulse Width High	3.5	3.9
t_{XCL}	Transmit Clock Pulse Width Low	4.1	4.5
t_{RCH}	Receive Clock Pulse Width High	2.5	
t_{RCL}	Receive Clock Pulse Width Low	2.5	
t_{DD}	Tx_data Output Delay	3.7	4.6
t_{DER}	Tx_Er Output Delay	3.7	4.6
t_{DVB}	Tx_Valid_Byte Output Delay	3.7	4.5
t_{DEN}	Tx_En Output Delay	3.2	4.7
t_{RDS}	Rx_data Setup Time	1.9	
t_{RDH}	Rx_data Hold Time	0	
t_{RVS}	Rx_Valid_Byte Setup Time	1.9	
t_{RVH}	Rx_Valid_Byte Hold Time	0	
t_{RES}	Rx_Er Setup Time	1.8	
t_{REH}	Rx_Er Hold Time	0	
t_{RDVS}	Rx_DV Setup Time	1.9	
t_{RDVH}	Rx_DV Hold Time	0	

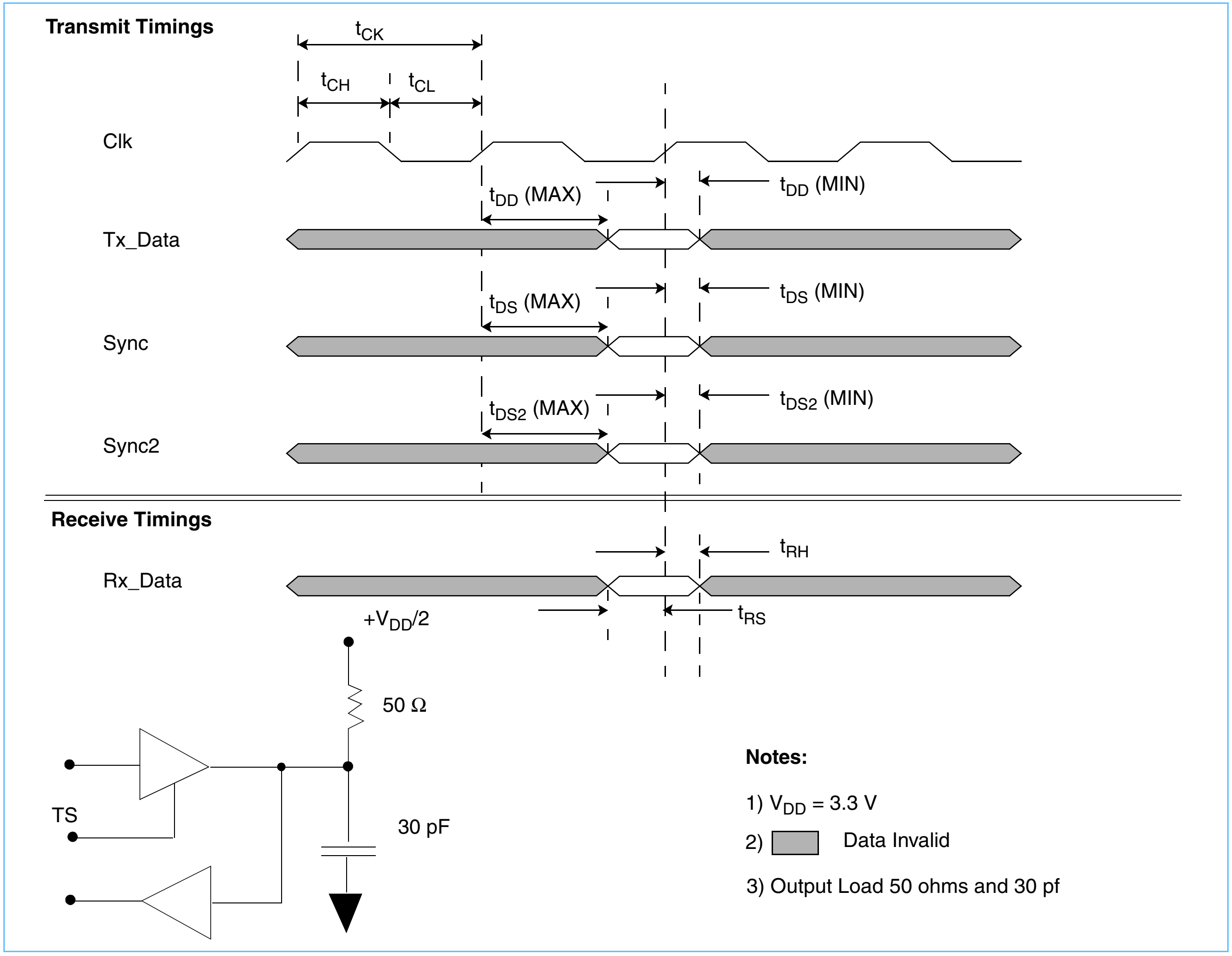
1. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

2.1.5.3 SMII Bus Pins

Table 2-23. 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
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
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
Sync2	Logically identical to Sync and provided for fanout purposes.	Output 5.0 V-tolerant 3.3 V LVTTL
$\overline{\text{CPDetect}}$	This pin must be pulled low to indicate when a Control Point is active on this DMU interface. Otherwise, this signal should be pulled high (inactive). The state of the signal is captured in the Data Mover Unit (DMU) Configuration (see *13.24 Data Mover Unit (DMU) Configuration Registers* on page 485) when the $\overline{\text{Reset}}$ signal is deasserted.	Input 5.0 V-tolerant 3.3 V LVTTL

Figure 2-10. SMII Timing Diagram



Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Table 2-24. SMII Timing Diagram Legend (for Figure 2-10)

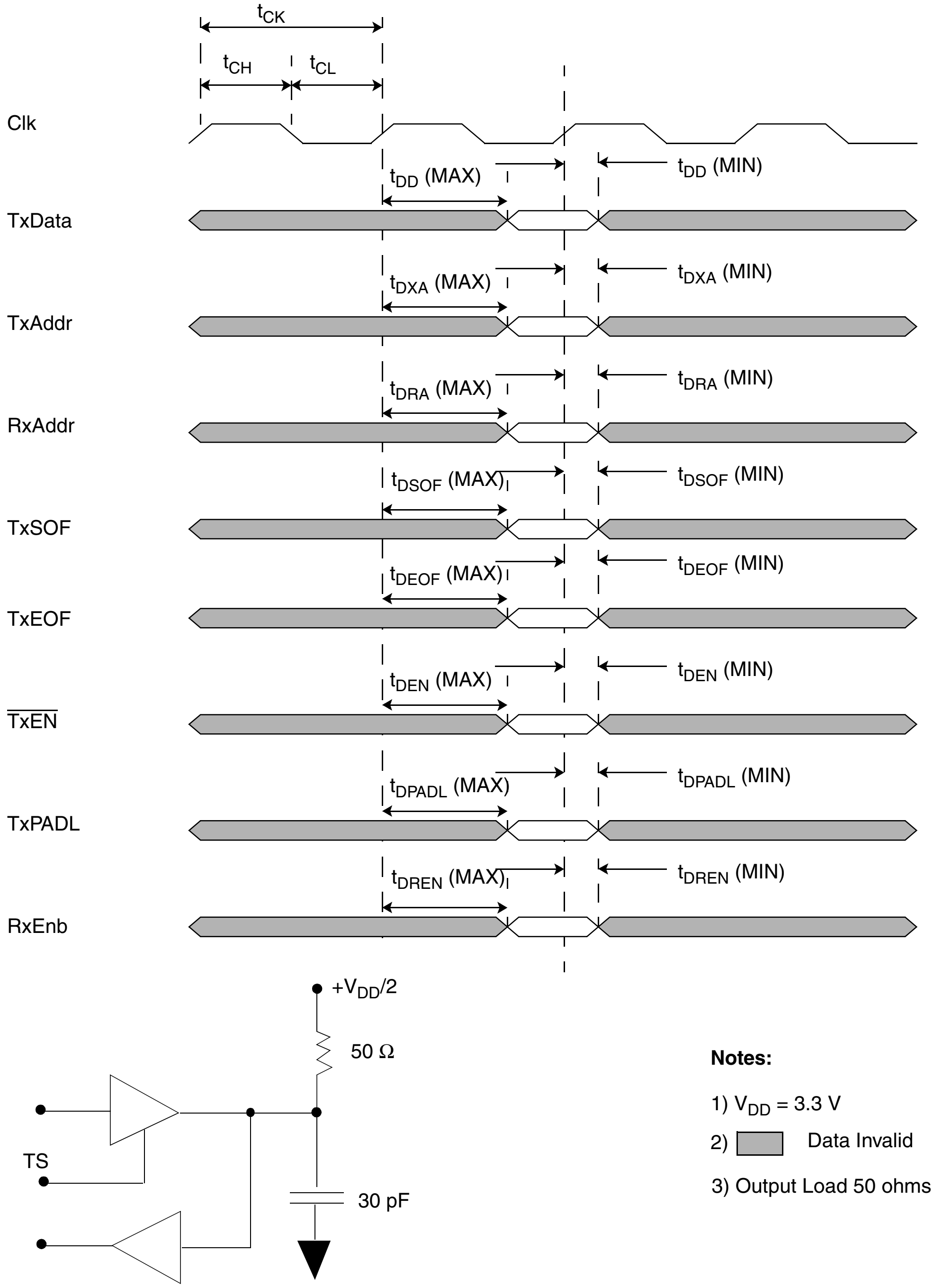
Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	Clk Cycle Time	8	
t_{CH}	Clk Pulse Width High	4	
t_{CL}	Clk Pulse Width Low	4	
t_{DD}	Tx_data_(9:0) Output Delay	1.9	4.7
tDS	Sync Output Delay	2.2	4.5
tDS2	Sync2 Output Delay	2.3	4.5
tRS	Rx_data_(9:0) Setup Time	0.8	
tRH	Rx_data_(9:0) Hold Time	0.2	

1. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

Table 2-25. 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
RxData(7:0) 8-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] is updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL
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
$\overline{\text{RxEnb}}$	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
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
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
RxVal	Receive valid data output indicates the receive signals RxData, RxEOF, and RxErr are valid from the framer. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL
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
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 $\overline{\text{TxEnb}}$. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL
$\overline{\text{TxEn}}$	Transmit write enable controls write access to the transmit interface. A framer port is selected on the falling edge of $\overline{\text{TxEnb}}$. Sampled on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL
TxAddr(1:0)	Transmit address bus uses $\overline{\text{TxEnb}}$ 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
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
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
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. 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
TxPFA	Transmit polled frame-available output indicates that the polled framer's transmit FIFO has free available space and the NP2G can write data into the framer's FIFO. Updated on the rising edge of Clk.	5.0 V-tolerant 3.3 V LVTTL

Figure 2-11. POS Transmit Timing Diagram



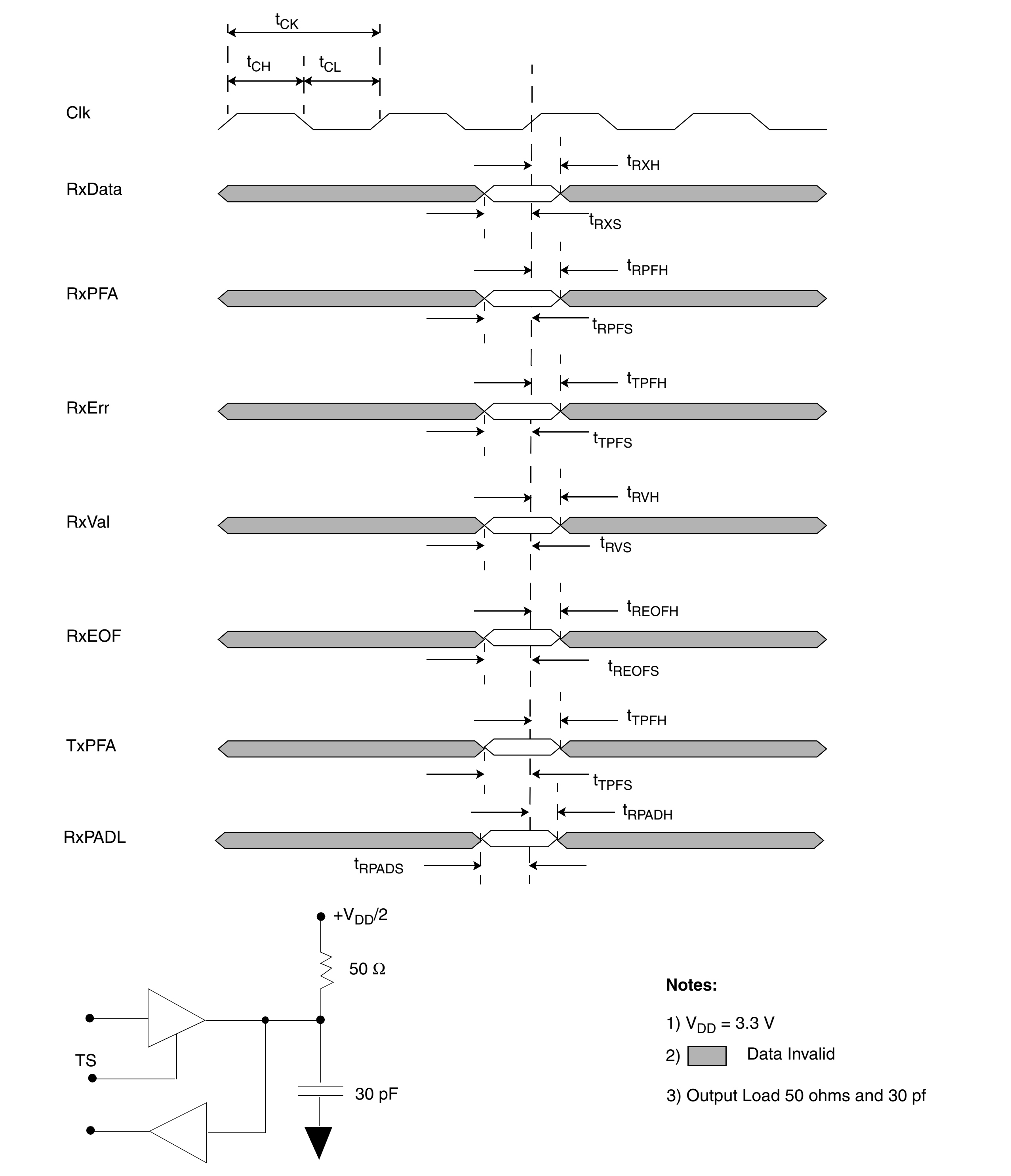
Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Figure 2-12. POS Receive Timing Diagram



Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Table 2-26. POS Timing Diagram Legend (for Figure 2-11 and Figure 2-12)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	Clk Cycle Time	10	
t_{CH}	Clk Clock Width High	2.1	
t_{CL}	Clk Clock Width Low	2.1	
t_{DD}	Tx_data_(31:0) Output Delay	2.2	4.6
t_{DXA}	Tx_ADDR_(1:0) Output Delay	2.2	4.6
t_{DRA}	Rx_ADDR_(1:0) Output Delay	2.1	4.3
t_{DSOF}	TxSOF Output Delay	2.3	4.6
t_{DEOF}	TxEOF Output Delay	2.2	4.5
t_{DEN}	$\overline{\text{TxEn}}$ Output Delay	1.9	4.7
t_{DPADL}	TxPADL_(1:0) Output Delay	2.2	4.6
t_{DREN}	RxEnb Output Delay	1.5	4.3
t_{RXS}	Rx_data_(31:0) Setup Time	1.9	
t_{RXH}	Rx_data_(31:0) Hold Time	0	
t_{RVS}	RxVal Setup Time	1.9	
t_{RVH}	RxVal Hold Time	0	
t_{RERS}	RxErr Setup Time	1.8	
t_{RERH}	RxErr Hold Time	0	
t_{REOFS}	RxEOF Setup Time	1.9	
t_{REOFH}	RxEOF Hold Time	0	
t_{RPFS}	RxPFA Setup Time	1.9	
t_{RPFH}	RxPFA Hold Time	0	
t_{TPFS}	TxPFA Setup Time	1.6	
t_{TPFH}	TxPFA Hold Time	0	
t_{RPADS}	RxPADL Setup Time	1.8	
t_{RPADH}	RxPADL Hold Time	0	

1. All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

2.1.6 PCI Pins

These pins interface with the PCI bus.

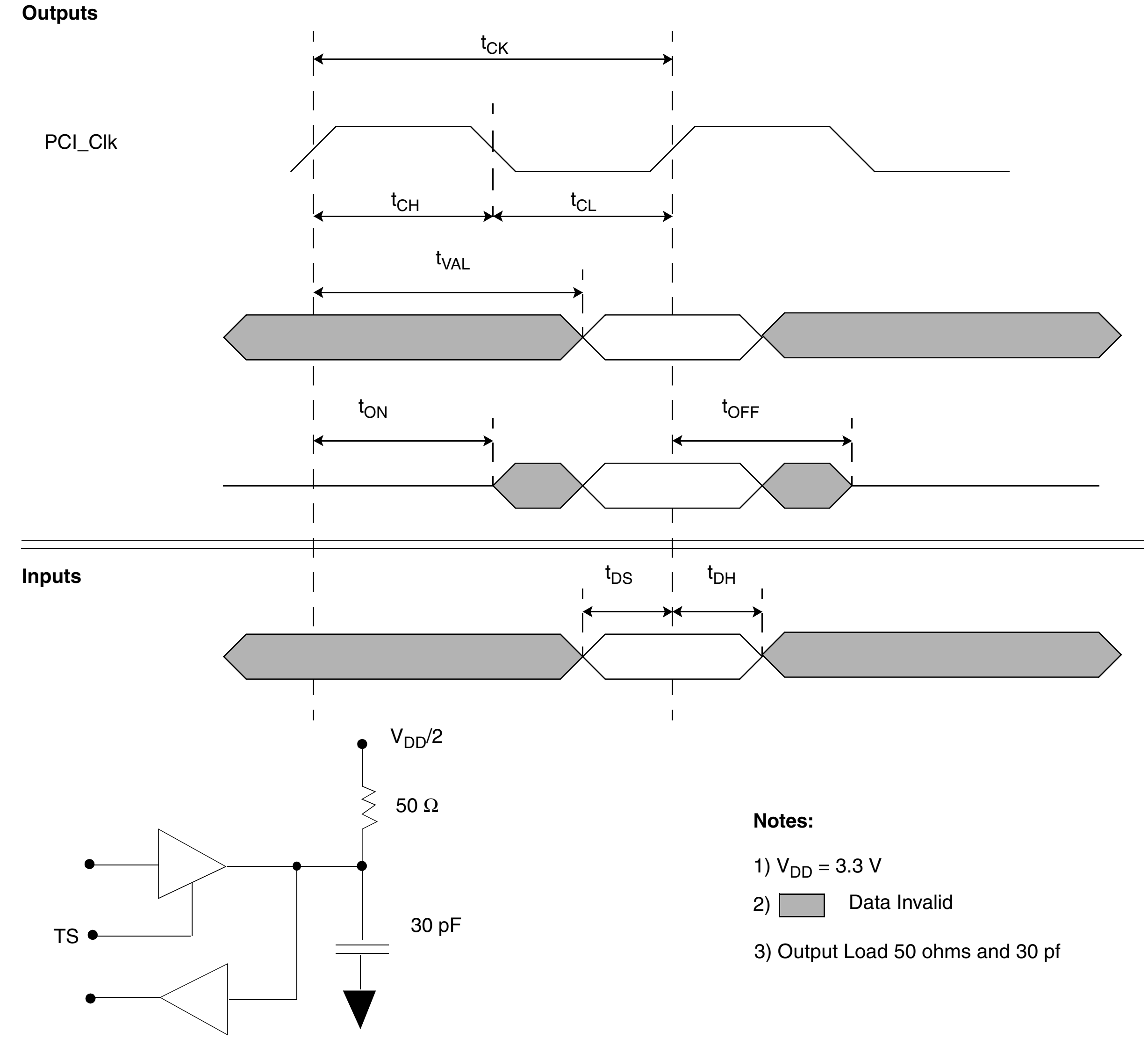
Table 2-27. PCI Pins (Page 1 of 2)

Signal	Description	Type
PCI_Clk	PCI Clock Signal (See PCI_Speed field below)	Input PCI (in)/ 3.3 V
PCI_AD(31:0)	PCI Multiplexed Address and Data Signals	Input/Output PCI (t/s) 3.3 V
$\overline{\text{PCI\_CBE}}$(3:0)	PCI Command/Byte Enable Signals	Input/Output PCI (t/s) 3.3 V
$\overline{\text{PCI\_Frame}}$	PCI Frame Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_IRdy}}$	PCI Initiator (Master) Ready Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_TRdy}}$	PCI Target (Slave) Ready Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_DevSel}}$	PCI Device Select Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_Stop}}$	PCI Stop Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_Request}}$	PCI Bus Request Signal	Output PCI (t/s) 3.3 V
$\overline{\text{PCI\_Grant}}$	PCI Bus Grant Signal	Input PCI (t/s) 3.3 V
PCI_IDSel	PCI Initialization Device Select Signal	Input PCI (in) 3.3 V
$\overline{\text{PCI\_PErr}}$	PCI Parity Error Signal	Input/Output PCI (s/t/s) 3.3 V
$\overline{\text{PCI\_SErr}}$	PCI System Error Signal	Input/Output PCI (o/d) 3.3 V
$\overline{\text{PCI\_IntA}}$	PCI Level Sensitive Interrupt	Output PCI (o/d) 3.3 V
PCI_Par	PCI Parity Signal. Covers all the data/address and the four command/BE signals.	Input/Output PCI (t/s) 3.3 V
Note: PCI I/Os are all configured for multi-point operation.		

Table 2-27. PCI Pins (Page 2 of 2)

Signal	Description	Type
PCI_Speed	PCI Speed. Controls Acceptable PCI Frequency Asynchronous Range by setting the PLB/PCI clock ratio. 0 PLB:PCI Mode 2:1. Acceptable PCI Frequency Asynchronous Range is 34.5 MHz to 66.6 MHz 1 PLB:PCI Mode 3:1. Acceptable PCI Frequency Asynchronous Range is 23.5 MHz to 44.5 MHz	Input 3.3 V-tolerant 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
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/Os are all configured for multi-point operation.		

Figure 2-13. PCI Timing Diagram



Notes:

1) V_{DD} = 3.3 V

2) ▨ Data Invalid

3) Output Load 50 ohms and 30 pf

Table 2-28. PCI Timing Diagram Legend (for Figure 2-13)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	PCI Cycle Time	15	
t_{CH}	Clk Clock Width High	7.5	
t_{CL}	Clk Clock Width Low	7.5	
t_{VAL}	Worst Case Output Delay	2.0	4.8
t_{ON}	PCI Bus Turn on Output Delay	2	
t_{OFF}	PCI Bus Turn off Output Delay		14
t_{DS}	Input Setup Time	2.4	
t_{DH}	Input Hold Time	0	
Note: All delays are measured with 1 ns slew time measured from 10-90% of input voltage.			

2.1.7 Management Bus Interface Pins

The signals from these pins are translated into various “host” buses by an external field-programmable gate array (FPGA) Serial/Parallel Manager (SPM).

Table 2-29. Management Bus Pins

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

Figure 2-14. SPM Bus Timing Diagram

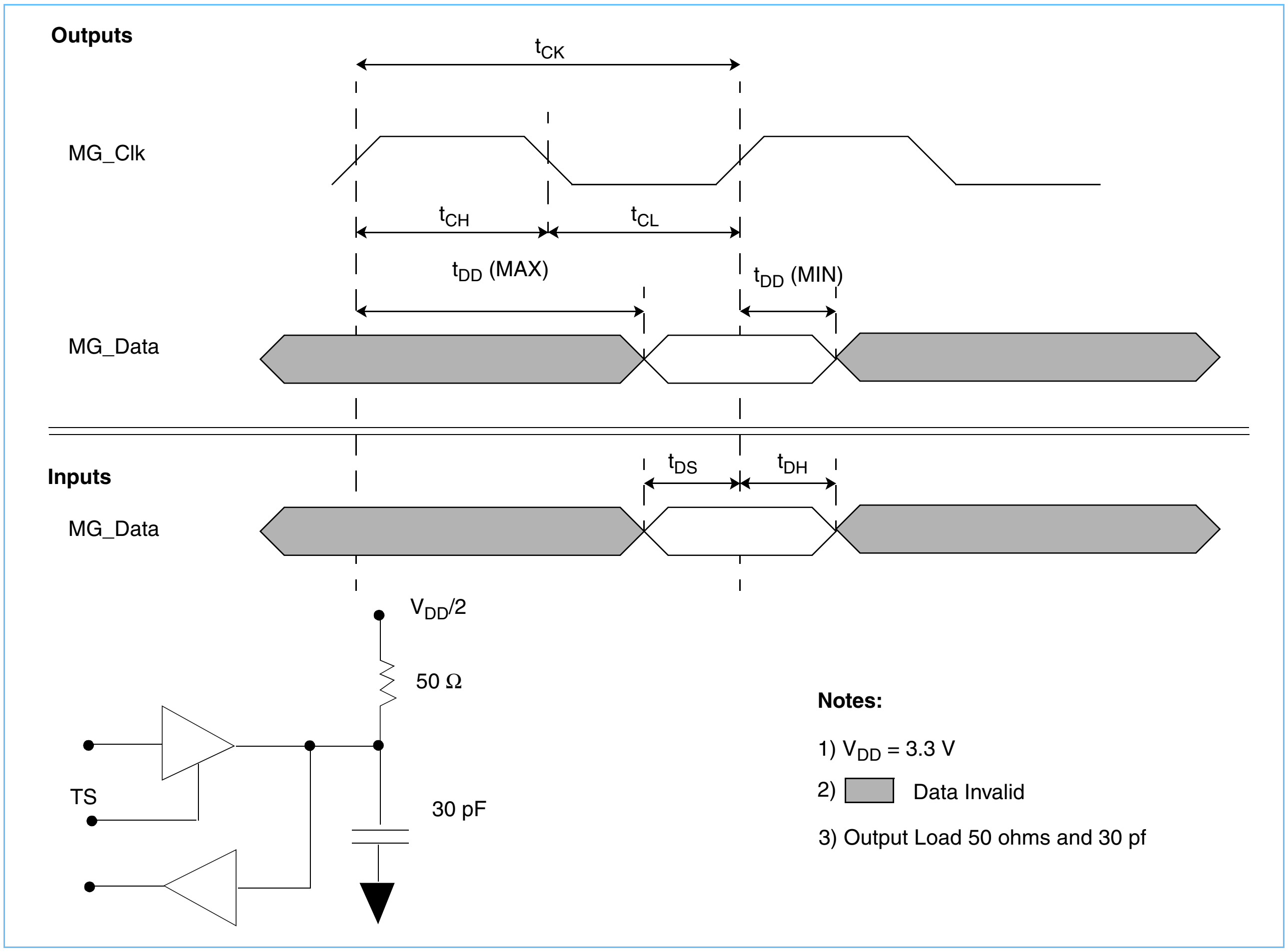


Table 2-30. SPM Bus Timing Diagram Legend (for Figure 2-14)

Symbol	Symbol Description	Minimum (ns)	Maximum (ns)
t_{CK}	SPM Cycle Time	30	
t_{CH}	Clock Pulse Width High	14.4	15.6
t_{CL}	Clock Pulse Width Low	14.4	15.6
t_{DD}	Data Output Delay	6.9	7.9
t_{DS}	Data Setup Time	5.1	
t_{DH}	Data Hold Time	0	

Note: mg_nintr is an asynchronous input and is not timed.
All delays are measured with 1 ns slew time measured from 10-90% of input voltage.

2.1.8 Miscellaneous Pins

Table 2-31. Miscellaneous Pins (Page 1 of 2)

Signal	Description	Type
IEW_Clock_A	The positive pin of an input differential pair. 50.6875 to 62.5 MHz. Generates IEW clock domains. Required to have cycle-to-cycle jitter ≤ ±150 ps. Duty Cycle tolerance must be ≤ ±10%. An on-chip differential terminator of 100 ohms is present between this pin and its complement pin.	IEW LVDS (see page 526)
IEW_Clock_A	The negative pin of an input differential pair. 50.6875 to 62.5 MHz.	Input LVDS (see page 526)
IEW_Clock_B	The positive pin of an input differential pair. 50.6875 to 62.5 MHz. Generates IEW clock domains. Required to have cycle-to-cycle jitter ≤ ±150 ps. Duty Cycle tolerance must be ≤ ±10%. An on-chip differential terminator of 100 ohms is present between this pin and its complement pin.	IEW LVDS (see page 526)
IEW_Clock_B	The negative pin of an input differential pair. 50.6875 to 62.5 MHz.	Input LVDS (see page 526)
Core_Clock	53.33 MHz oscillator - generates 266 /133 clock domains. Required to have cycle-to-cycle jitter ≤ ±150 ps. Duty Cycle tolerance must be ≤ ±10%.	Input 5.0 V-tolerant 3.3 V LVTTL
Clock125	125 MHz oscillator. Required to have cycle-to-cycle jitter ≤ ±60 ps. Duty Cycle tolerance must be ≤ ± 5%. This clock is required only when supporting TBI and GMII DMU bus modes.	Input 5.0 V-tolerant 3.3 V LVTTL
Reset	Reset NP2G - signal must be driven active low for a minimum of 1µs to ensure a proper reset of the NP2G. All input clocks (IEW_Clock_A, IEW_Clock_A, IEW_Clock_B, IEW_Clock_B, Core_Clock, Clock125 if in use, and PCI_Clk) must be running prior to the activation of this signal.	Input 5.0 V-tolerant 3.3 V LVTTL
Operational	NP2G operational - pin is driven active low when both the NP2G Ingress and Egress Macros have completed their initialization. It remains active until a subsequent Reset is issued.	Output (o/d) 5.0 V-tolerant 3.3 V LVTTL
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 debug bus (DMU_D) output to be gated from the probe. 10 JTAG Test Mode 11 LSSD Test Mode	Input CMOS 1.8 V
JTAG_TRst	JTAG test reset. For normal functional operation, this pin must be connected to the same card source that is connected to the 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
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
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
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
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

Table 2-31. Miscellaneous Pins (Page 2 of 2)

Signal	Description	Type
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 phase-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 *2.1.9 PLL Filter Circuit on page 80* 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 2-16: PLL Filter Circuit Diagram* on page 80.	Input PLL_GND 0.0 V
Thermal_In	Input pad of the thermal monitor (resistor). See *2.1.10 Thermal I/O Usage* on page 80 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 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
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
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
$\overline{\text{PIO(2:0)}}$	Programmable I/O	Input/Output CMOS 2.5 V
PGM_GND PGM_VDD	These IO are used for chip manufacturing and test purposes and should be left unconnected on the card.	-

The termination network details for the signals are the same as those for the NP4GS3. See the guidelines contained in the *IBM PowerNP NP4GS3 PC Card Layout Guidelines*.

Table 2-32. 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
Reserved_IO (17:18)		Pull down to GND
Reserved_IO (19:21, 22, 23:26, 28:49, 54, 55, 56)		Pull down to GND[1]
Reserved_IO (50:53)		Pull-up to 3.3 V[1]
RES_Data		Pull-down and do not connect with other signals
RES_Sync		Pull-down and do not connect with other signals
Signals requiring a DC connection that varies across different applications		
PCI_Speed		Choose up or down based on system PCI bus speed
MG_nIntr		Pull-down if no system device drives this signal
MG_Data		Pull-down when external SPM module is attached
Boot_Picocode		Choose up or down based on picocode load location
Boot_PPC		Choose up or down based on PPC code load location
LU_R_$\overline{\text{WRT}}$, SCH_R_$\overline{\text{WRT}}$		Pull up when system diagnostics uses ZBT SRAM to hold information through a software controlled reset (see *Software Controlled Reset Register (Soft_Reset)* on page 455).
Signals that have an AC connection, but also require pull-up or pull-down		
$\overline{\text{Operational}}$		Pull-up
DMU_A(0), DMU_C(0), DMU_D(0)	$\overline{\text{CPDetect}}$	If control point then pull-down, otherwise pull-up
DMU_A(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_C(4), DMU_D(4)	DMU in any POS mode. RxVal	Pull-down
D3_DQS(1:0), D2_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

1. Up to eight I/Os that are pulled to the same voltage can be connected to a single resistor if its value is close to 300 ohms. If individual resistors are used, they may be up to 1000 ohms.

Note: In addition to the signals in this table, any interface signals not used for a particular application should be pulled to their inactive state for control signals or either state for data signals. This prevents extraneous switching of the circuitry, which can cause current surges and affect other signals.

Table 2-33. Pins Requiring Connections to Other Pins

First Grid Location	Second Grid Location	Third Grid Location	Notes
AK01	Y01		1
AJ02	W02		1
AF01	W03		1
AE02	AA01		1
AH01	AB01		1
AH03	AA02		1
AE01	AC06		1
AE03	AC05		1
AD03	AC07		1
AD01	AD05		1
AC02	AE04		1
AC03	AE05		1
AB03	AF03		1
AA04	AF05		1
AA03	AG04		1
Y03	AG02		1
E02	R02		
D01	P01		
J02	N01		
H01	R03		
F03	N02		
F01	M01		
G02	P03		
G04	N03		
J03	L05		
J01	L06		
K01	K05		
K03	L07		
L03	J05		
L02	J04		
N04	H05		
M03	H03		
V19	U19	W20	
AG27	AE25	T19	
Note: All pins in the same row should be connected together on the board. 1. If wiring congestion occurs, the pins in these rows can be rewired as shown in *Table 2-34* and *Figure 2-15*.			

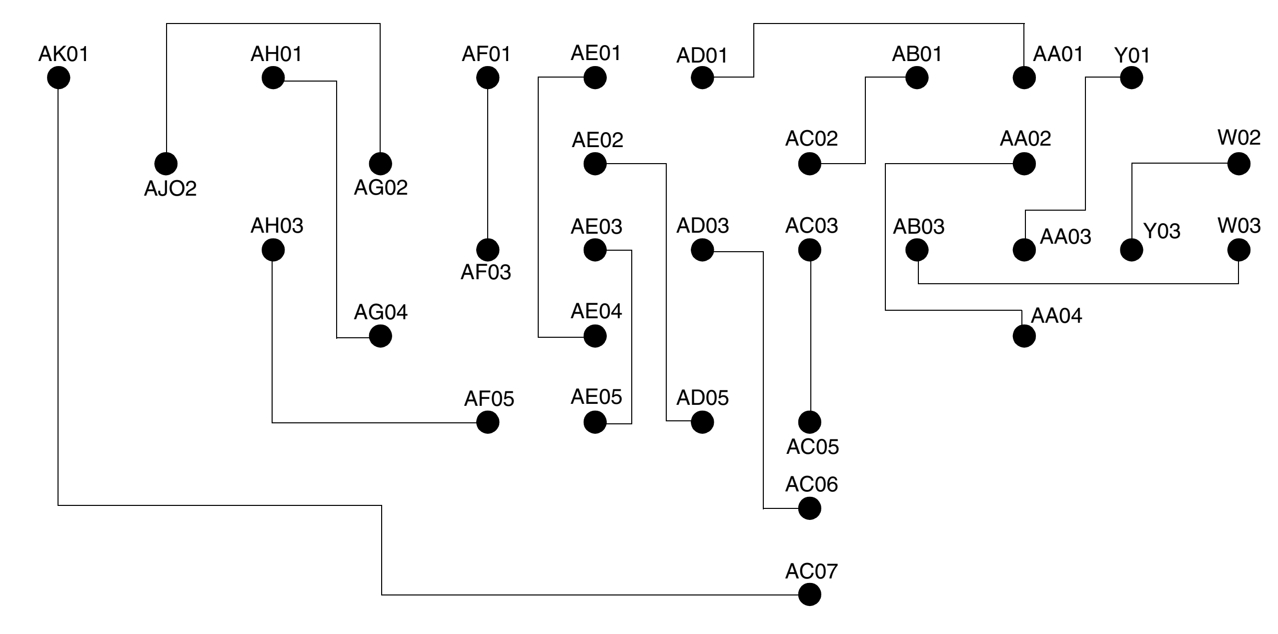
The alternate wiring scheme illustrated in *Table 2-34* and *Figure 2-15* can be used if wiring congestion occurs.

Note: This scheme applies only to a subset of the pins requiring connections to other pins (see *Table 2-33*).

Table 2-34. Alternate Wiring for Pins Requiring Connections to Other Pins

First Grid Location	Second Grid Location
AK01	AC07
AJ02	AG02
AF01	AF03
AE02	AD05
AH01	AG04
AH03	AF05
AE01	AE04
AE03	AE05
AD03	AC06
AD01	AA01
AC02	AB01
AC03	AC05
AB03	W03
AA04	AA02
AA03	Y01
Y03	W02

Figure 2-15. Alternate Wiring for Pins Requiring Connection to Other Pins



2.1.9 PLL Filter Circuit

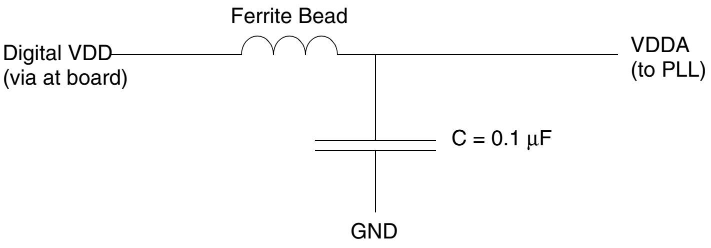
V_{DDA} is the voltage supply pin to the analog circuits in the PLL. Noise on V_{DDA} causes phase jitter at the output of the PLL. V_{DDA} is brought to a package pin to isolate it from the noisy internal digital V_{DD} signal. If little noise is expected at the board level, then V_{DDA} can be connected 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.

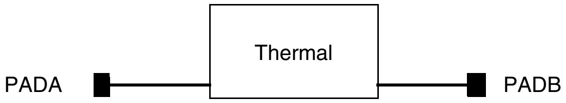
Figure 2-16. PLL Filter Circuit Diagram



2.1.10 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 1500 + 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 81.

Figure 2-17. Thermal Monitor



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

2.1.10.1 Temperature Calculation

The chip temperature can be calculated from

$$T_{\text{chip}} = \frac{1}{t_c\,(R_{\text{measured}} - R_{\text{calibrated}})} + T_{\text{calibrated}}\ {}^\circ C$$

where:

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

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

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

2.1.10.2 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 device is dissipating little (less than 100 mW) or no power. Apply I_{dc} and wait for a fixed time t_m, where t_m = approximately 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,

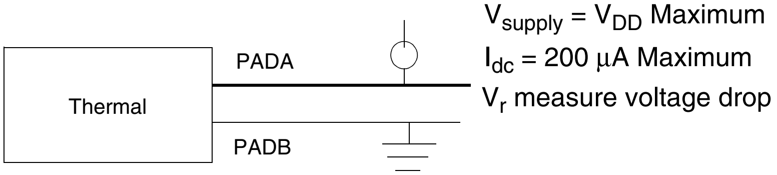
$$Tc = \left[\frac{\Delta Vr}{Idc \times \Delta T}\right]\frac{\Omega}{{}^\circ C} \qquad ,$$

where:

Δ T = temperature change, °C

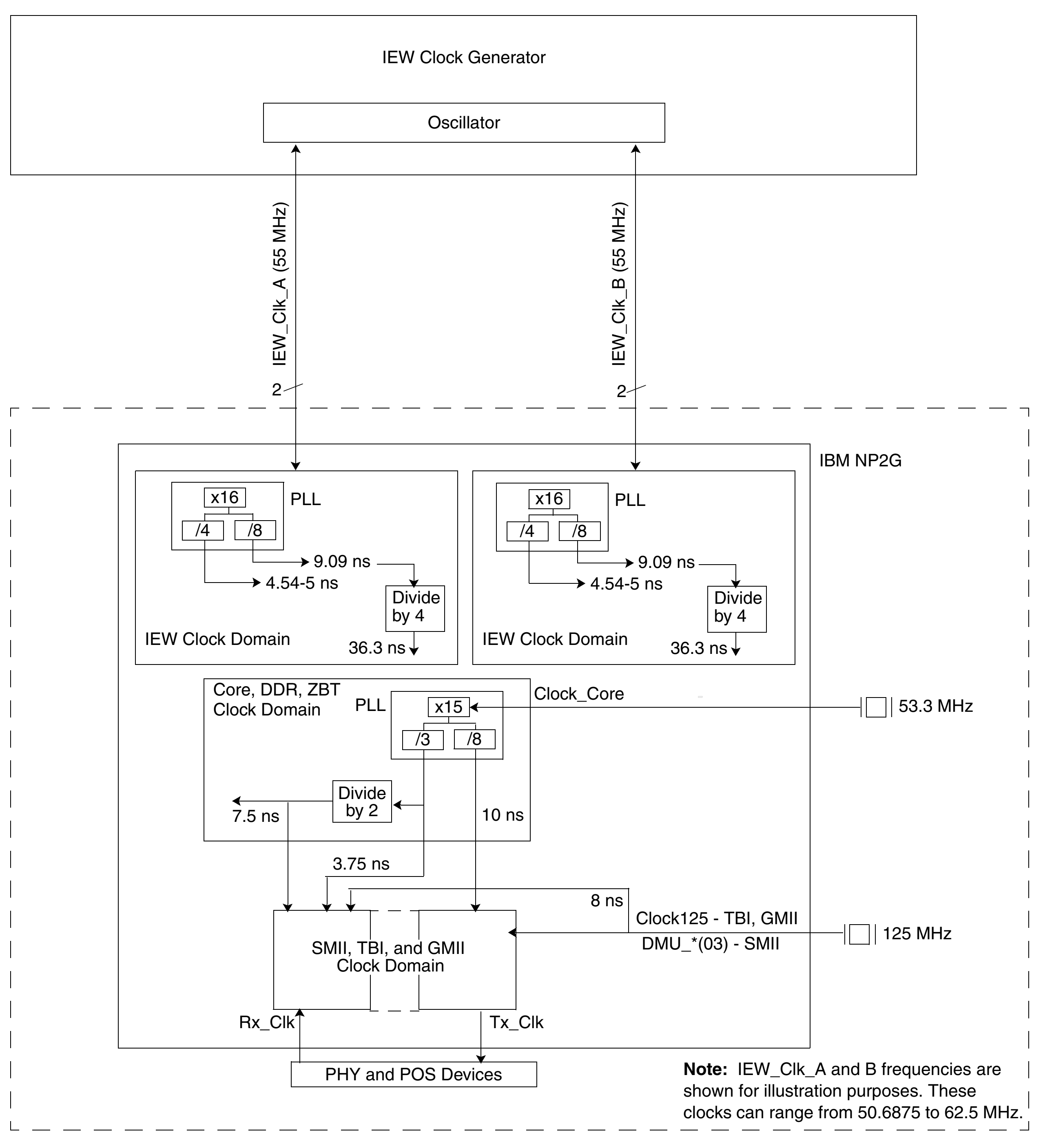
Δ V_r = voltage drop, V

I_{dc} = applied current, A


Thermal
PADA
PADB
V_{supply} = V_{DD} Maximum
I_{dc} = 200 µA Maximum
V_r measure voltage drop

2.2 Clocking Domains

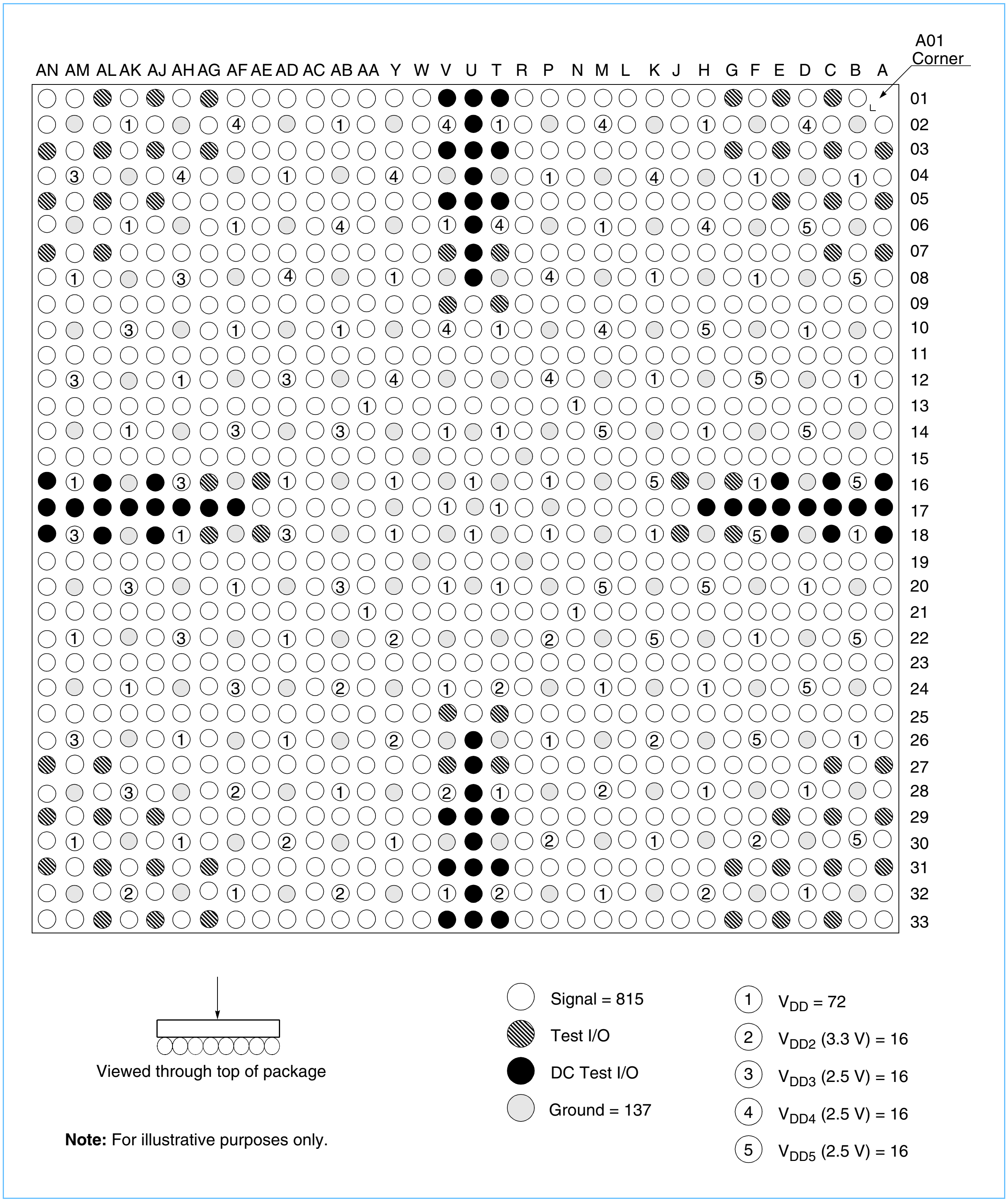
Figure 2-18. Clock Generation and Distribution



Note: IEW_Clk_A and B frequencies are shown for illustration purposes. These clocks can range from 50.6875 to 62.5 MHz.

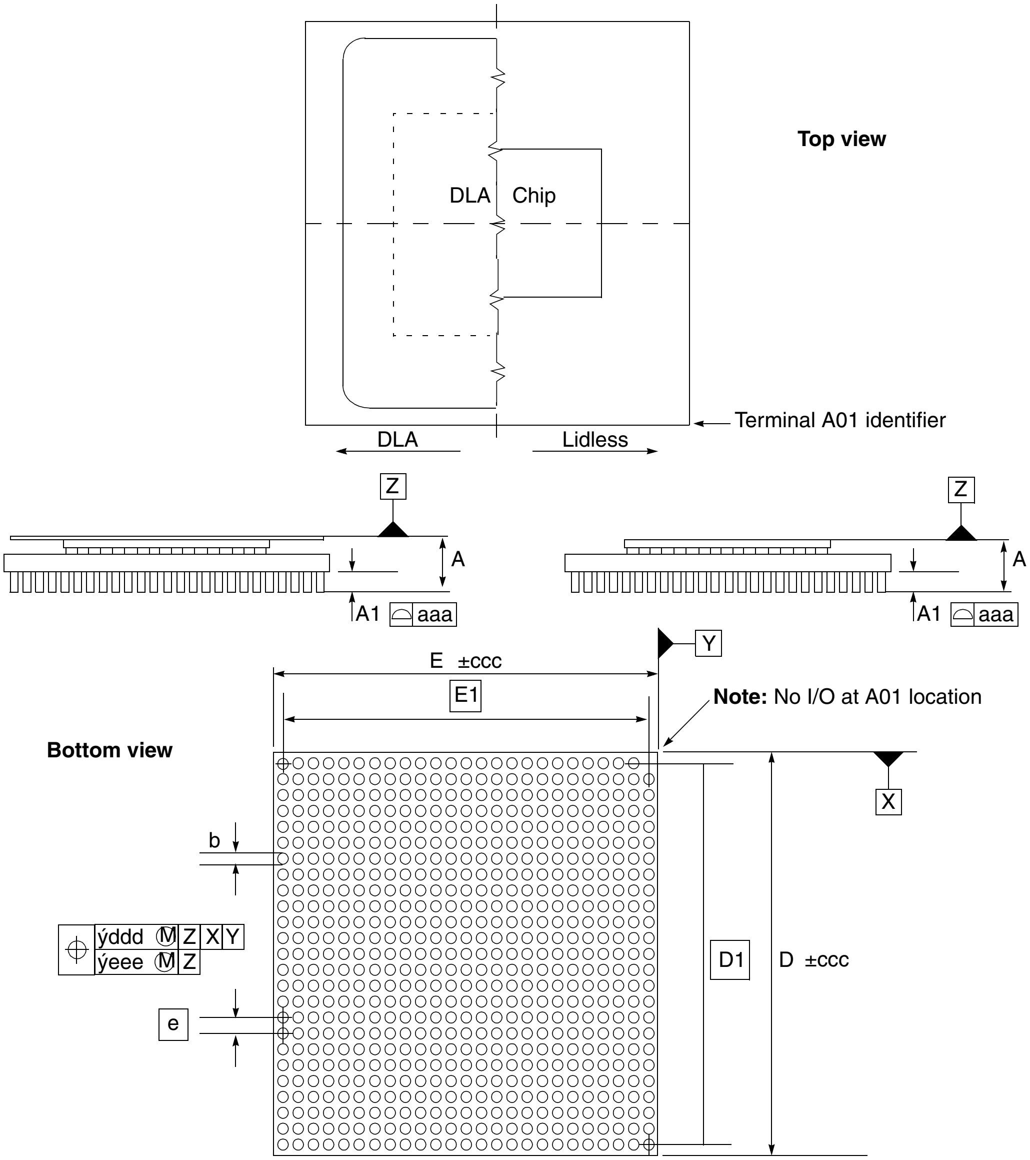
See *NP2G DMU Bus Clock Connections* on page 58 and *NP2G DMU Bus Clock Connections (POS Overview)* on page 59 for related clock information.

Figure 2-19. Pins Diagram



2.3 Mechanical Specifications

Figure 2-20. Mechanical Diagram



Notes:

1. Refer to *Table 2-35* on page 85 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 2-35. 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 one I/O at the A01 corner of the array.

2.4 IEEE 1149 (JTAG) Compliance

2.4.1 Statement of JTAG Compliance

The NP2G 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 2-36*.

Table 2-36. 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 NP2G's design. *Table 2-37* documents all implemented public instructions.

Table 2-37. 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. Device 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.

2.5 Signal Pin Lists

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position
Boot_Picocode	K07
Boot_PPC	L04
$\overline{\text{C405\_Debug\_Halt}}$	AB23
cam_cp_response(00)	U09
cam_cp_response(01)	T11
cam_cp_response(02)	P05
cam_cp_response(03)	R06
cam_cp_response(04)	T13
cam_cp_response(05)	P07
cam_cp_response(06)	T05
cam_cp_response(07)	R01
cam_cp_response(08)	R10
cam_cp_response(09)	P09
cam_cp_response(10)	R09
cam_cp_response(11)	R14
cam_cp_response(12)	R13
cam_cp_response(13)	N07
Clock125	B33
Core_Clock	C33
D0_Addr(00)	AL10
D0_Addr(01)	AN10
D0_Addr(02)	AJ09
D0_Addr(03)	AN06
D0_Addr(04)	AA14
D0_Addr(05)	AB15
D0_Addr(06)	AM07
D0_Addr(07)	AL08
D0_Addr(08)	AM11
D0_Addr(09)	AL11
D0_Addr(10)	AC14
D0_Addr(11)	AG10
D0_Addr(12)	AF11
$\overline{\text{D0\_CS}}$	AN09
D0_Data(00)	AA12
D0_Data(01)	AD09
D0_Data(02)	AG07
D0_Data(03)	AB11
D0_Data(04)	AN02
D0_Data(05)	AM03
D0_Data(06)	AN01
D0_Data(07)	AN03
D0_Data(08)	AL04
D0_Data(09)	AG03
D0_Data(10)	AH07
D0_Data(11)	AE09
D0_Data(12)	AL03
D0_Data(13)	AC11
D0_Data(14)	AJ06
D0_Data(15)	AK05
D0_Data(16)	AJ07
D0_Data(17)	AN04
D0_Data(18)	AN07
D0_Data(19)	AL07
D0_Data(20)	AF09
D0_Data(21)	AG08
D0_Data(22)	AE10
D0_Data(23)	AD11
D0_Data(24)	AN08
D0_Data(25)	AL09
D0_Data(26)	AL06
D0_Data(27)	AM05
D0_Data(28)	AB13
D0_Data(29)	AC15
D0_Data(30)	AK07
D0_Data(31)	AJ08
D0_DQS(0)	AG09
D0_DQS(1)	AC12
D0_DQS(2)	AE11
D0_DQS(3)	AH09
$\overline{\text{D0\_WE}}$	AM09
D2_Addr(00)	AJ22
D2_Addr(01)	AH21
D2_Addr(02)	AN21
D2_Addr(03)	AM21
D2_Addr(04)	AH23
D2_Addr(05)	AC20
D2_Addr(06)	AD21
D2_Addr(07)	AG23
D2_Addr(08)	AN22
D2_Addr(09)	AL21
D2_Addr(10)	AK23
D2_Addr(11)	AJ23
D2_Addr(12)	AA19
$\overline{\text{D2\_CS}}$	AL22
D2_Data(00)	AN18
D2_Data(01)	AJ19
D2_Data(02)	W18
D2_Data(03)	AA18
D2_Data(04)	AJ20
D2_Data(05)	AL20
D2_Data(06)	AN19
D2_Data(07)	AE20
D2_Data(08)	AE17
D2_Data(09)	AE21
D2_Data(10)	AG22
D2_Data(11)	AN20
D2_Data(12)	AM19
D2_Data(13)	AK21
D2_Data(14)	AJ21
D2_Data(15)	Y19
D2_DQS(0)	AB19
D2_DQS(1)	AK25

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position
$\overline{\text{D2\_WE}}$	AJ24
D3_Addr(00)	AG19
D3_Addr(01)	AJ16
D3_Addr(02)	AF17
D3_Addr(03)	AJ17
D3_Addr(04)	AG18
D3_Addr(05)	AE18
D3_Addr(06)	AA17
D3_Addr(07)	AC18
D3_Addr(08)	AK19
D3_Addr(09)	AL19
D3_Addr(10)	AN17
D3_Addr(11)	AK17
D3_Addr(12)	AE19
$\overline{\text{D3\_CS}}$	AL18
D3_Data(00)	AG13
D3_Data(01)	AA16
D3_Data(02)	AG14
D3_Data(03)	AE15
D3_Data(04)	AL17
D3_Data(05)	AM17
D3_Data(06)	AL15
D3_Data(07)	AK15
D3_Data(08)	W17
D3_Data(09)	AE16
D3_Data(10)	AG16
D3_Data(11)	AH17
D3_Data(12)	AG17
D3_Data(13)	AG15
D3_Data(14)	AF15
D3_Data(15)	AF19
D3_DQS(0)	AG20
D3_DQS(1)	AC17
$\overline{\text{D3\_WE}}$	AD19
D4_Addr(00)	C12
D4_Addr(01)	A11
D4_Addr(02)	J12
D4_Addr(03)	H11
D4_Addr(04)	G10
D4_Addr(05)	L13
D4_Addr(06)	C11
D4_Addr(07)	B11
D4_Addr(08)	D09
D4_Addr(09)	C08
D4_Addr(10)	M13
D4_Addr(11)	N14
D4_Addr(12)	B07
$\overline{\text{D4\_CS}}$	E10
D4_Data(00)	M17
D4_Data(01)	L16
D4_Data(02)	D15
D4_Data(03)	C15
D4_Data(04)	A17
D4_Data(05)	D17
D4_Data(06)	H09
D4_Data(07)	G14
D4_Data(08)	G13
D4_Data(09)	K15
D4_Data(10)	C16
D4_Data(11)	A16
D4_Data(12)	E15
D4_Data(13)	F15
D4_Data(14)	R17
D4_Data(15)	N16
D4_Data(16)	E14
D4_Data(17)	C14
D4_Data(18)	E09
D4_Data(19)	A15
D4_Data(20)	L15
D4_Data(21)	J14
D4_Data(22)	G12
D4_Data(23)	H13
D4_Data(24)	A14
D4_Data(25)	B15
D4_Data(26)	D13
D4_Data(27)	E13
D4_Data(28)	P15
D4_Data(29)	E12
D4_Data(30)	F13
D4_Data(31)	A13
D4_DQS(0)	D11
D4_DQS(1)	E11
D4_DQS(2)	N15
D4_DQS(3)	M15
$\overline{\text{D4\_WE}}$	F11
D6_Addr(00)	AL28
D6_Addr(01)	AN26
D6_Addr(02)	AE24
D6_Addr(03)	AG26
D6_Addr(04)	AF25
D6_Addr(05)	AL27
D6_Addr(06)	AN30
D6_Addr(07)	AJ27
D6_Addr(08)	AK29
D6_Addr(09)	AJ28
D6_Addr(10)	AL29
D6_Addr(11)	AG21
D6_Addr(12)	AH27
D6_ByteEn(0)	AG24
D6_ByteEn(1)	AF23
$\overline{\text{D6\_CS}}$	AL31
D6_Data(00)	AL23
D6_Data(01)	AM23
D6_Data(02)	AL26
D6_Data(03)	AM27

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position
D6_Data(04)	AB21
D6_Data(05)	AN28
D6_Data(06)	AN24
D6_Data(07)	AL24
D6_Data(08)	AH25
D6_Data(09)	AE23
D6_Data(10)	AC21
D6_Data(11)	AN25
D6_Data(12)	AJ26
D6_Data(13)	AK27
D6_Data(14)	AE22
D6_Data(15)	AC22
D6_DQS(0)	AL30
D6_DQS(1)	AN31
D6_DQS(2)	AN33
D6_DQS(3)	AM31
D6_DQS_Par(00)	AN32
D6_DQS_Par(01)	AF21
D6_Parity(00)	AM29
D6_Parity(01)	AN23
$\overline{\text{D6\_WE}}$	AJ25
DA_BA(0)	AN29
DA_BA(1)	AA20
$\overline{\text{DA\_CAS}}$	AN27
DA_$\overline{\text{Clk}}$	AL25
DA_Clk	AM25
$\overline{\text{DA\_RAS}}$	AG25
DB_BA(0)	AG11
DB_BA(1)	AD13
$\overline{\text{DB\_CAS}}$	AJ12
DB_Clk	AH19
$\overline{\text{DB\_Clk}}$	AJ18
$\overline{\text{DB\_RAS}}$	AE13
DC_BA(0)	D25
DC_BA(1)	J17
$\overline{\text{DC\_CAS}}$	N19
DC_$\overline{\text{Clk}}$	D23
DC_Clk	E23
$\overline{\text{DC\_RAS}}$	C21
DD_BA(0)	A12
DD_BA(1)	J13
$\overline{\text{DD\_CAS}}$	G11
DD_$\overline{\text{Clk}}$	B13
DD_Clk	C13
$\overline{\text{DD\_RAS}}$	K13
DE_BA(0)	AM01
DE_BA(1)	AH05
$\overline{\text{DE\_CAS}}$	AL02
DE_Clk	AJ04
DE_$\overline{\text{Clk}}$	AJ05
$\overline{\text{DE\_RAS}}$	AK03
DMU_A(00)	V31
DMU_A(01)	V29
DMU_A(02)	V27
DMU_A(03)	W33
DMU_A(04)	W32
DMU_A(05)	W31
DMU_A(06)	W30
DMU_A(07)	W29
DMU_A(08)	W28
DMU_A(09)	W27
DMU_A(10)	W26
DMU_A(11)	W25
DMU_A(12)	W24
DMU_A(13)	W23
DMU_A(14)	Y33
DMU_A(15)	Y31
DMU_A(16)	Y29
DMU_A(17)	Y27
DMU_A(18)	Y25
DMU_A(19)	Y23
DMU_A(20)	AA33
DMU_A(21)	AA32
DMU_A(22)	AA31
DMU_A(23)	AA30
DMU_A(24)	AA29
DMU_A(25)	AA28
DMU_A(26)	AA27
DMU_A(27)	AA26
DMU_A(28)	AA25
DMU_A(29)	AB33
DMU_A(30)	AB31
DMU_C(00)	L33
DMU_C(01)	L32
DMU_C(02)	L31
DMU_C(03)	L30
DMU_C(04)	L29
DMU_C(05)	L28
DMU_C(06)	L27
DMU_C(07)	L26
DMU_C(08)	L25
DMU_C(09)	L24
DMU_C(10)	L23
DMU_C(11)	M33
DMU_C(12)	M31
DMU_C(13)	M29
DMU_C(14)	M27
DMU_C(15)	M25
DMU_C(16)	M23
DMU_C(17)	N33
DMU_C(18)	N32
DMU_C(19)	N31
DMU_C(20)	N30
DMU_C(21)	N29
DMU_C(22)	N28

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
DMU_C(23)	N27	DMU_D(27)	K31	DS0_Data(17)	G26
DMU_C(24)	N26	DMU_D(28)	K29	DS0_Data(18)	J24
DMU_C(25)	N25	DMU_D(29)	E31	DS0_Data(19)	K23
DMU_C(26)	N24	DMU_D(30)	G31	DS0_Data(20)	A26
DMU_C(27)	N23	DS0_Addr(00)	A28	DS0_Data(21)	C25
DMU_C(28)	P33	DS0_Addr(01)	N20	DS0_Data(22)	C28
DMU_C(29)	P31	DS0_Addr(02)	M19	DS0_Data(23)	B29
DMU_C(30)	P29	DS0_Addr(03)	B27	DS0_Data(24)	M21
DMU_D(00)	D33	DS0_Addr(04)	C26	DS0_Data(25)	L19
DMU_D(01)	D31	DS0_Addr(05)	B23	DS0_Data(26)	D27
DMU_D(02)	G28	DS0_Addr(06)	C23	DS0_Data(27)	E26
DMU_D(03)	J29	DS0_Addr(07)	G24	DS0_Data(28)	A25
DMU_D(04)	E30	DS0_Addr(08)	H23	DS0_Data(29)	B25
DMU_D(05)	F33	DS0_Addr(09)	J22	DS0_Data(30)	G25
DMU_D(06)	F31	DS0_Addr(10)	A23	DS0_Data(31)	L22
DMU_D(07)	F29	DS0_Addr(11)	C22	DS0_DQS(0)	F25
DMU_D(08)	G32	DS0_Addr(12)	E24	DS0_DQS(1)	C24
DMU_D(09)	K25	$\overline{\text{DS0\_CS}}$	A31	DS0_DQS(2)	A24
DMU_D(10)	G30	DS0_Data(00)	P19	DS0_DQS(3)	E25
DMU_D(11)	G29	DS0_Data(01)	A32	$\overline{\text{DS0\_WE}}$	J23
DMU_D(12)	E32	DS0_Data(02)	B31	DS1_Addr(00)	N17
DMU_D(13)	H33	DS0_Data(03)	A33	DS1_Addr(01)	J18
DMU_D(14)	H31	DS0_Data(04)	C30	DS1_Addr(02)	G18
DMU_D(15)	H29	DS0_Data(05)	C31	DS1_Addr(03)	E17
DMU_D(16)	H27	DS0_Data(06)	F27	DS1_Addr(04)	H17
DMU_D(17)	J33	DS0_Data(07)	H21	DS1_Addr(05)	G19
DMU_D(18)	J32	DS0_Data(08)	C29	DS1_Addr(06)	H19
DMU_D(19)	J31	DS0_Data(09)	A29	DS1_Addr(07)	H15
DMU_D(20)	J30	DS0_Data(10)	E28	DS1_Addr(08)	G15
DMU_D(21)	K27	DS0_Data(11)	D29	DS1_Addr(09)	G17
DMU_D(22)	J28	DS0_Data(12)	E27	DS1_Addr(10)	F17
DMU_D(23)	J27	DS0_Data(13)	A30	DS1_Addr(11)	G16
DMU_D(24)	J26	DS0_Data(14)	A27	DS1_Addr(12)	J16
DMU_D(25)	J25	DS0_Data(15)	C27	$\overline{\text{DS1\_CS}}$	E16
DMU_D(26)	K33	DS0_Data(16)	H25	DS1_Data(00)	A22

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
DS1_Data(01)	G23	$\overline{\text{DS1\_WE}}$	C17	GND	AK08
DS1_Data(02)	K21	DUM	A01	GND	AK12
DS1_Data(03)	L20	GND	AB04	GND	AK16
DS1_Data(04)	F23	GND	AB08	GND	AK18
DS1_Data(05)	B21	GND	AB12	GND	AK22
DS1_Data(06)	A21	GND	AB16	GND	AK26
DS1_Data(07)	F21	GND	AB18	GND	AK30
DS1_Data(08)	E22	GND	AB22	GND	AM02
DS1_Data(09)	L18	GND	AB26	GND	AM06
DS1_Data(10)	L17	GND	AB30	GND	AM10
DS1_Data(11)	E21	GND	AD02	GND	AM14
DS1_Data(12)	D21	GND	AD06	GND	AM20
DS1_Data(13)	B19	GND	AD10	GND	AM24
DS1_Data(14)	A20	GND	AD14	GND	AM28
DS1_Data(15)	G22	GND	AD20	GND	AM32
DS1_Data(16)	J21	GND	AD24	GND	B02
DS1_Data(17)	J15	GND	AD28	GND	B06
DS1_Data(18)	J20	GND	AD32	GND	B10
DS1_Data(19)	A19	GND	AF04	GND	B14
DS1_Data(20)	E18	GND	AF08	GND	B20
DS1_Data(21)	C20	GND	AF12	GND	B24
DS1_Data(22)	E20	GND	AF16	GND	B28
DS1_Data(23)	N18	GND	AF18	GND	B32
DS1_Data(24)	F19	GND	AF22	GND	D04
DS1_Data(25)	E19	GND	AF26	GND	D08
DS1_Data(26)	A18	GND	AF30	GND	D12
DS1_Data(27)	C18	GND	AH02	GND	D16
DS1_Data(28)	K19	GND	AH06	GND	D18
DS1_Data(29)	G21	GND	AH10	GND	D22
DS1_Data(30)	G20	GND	AH14	GND	D26
DS1_Data(31)	J19	GND	AH20	GND	D30
DS1_DQS(0)	B17	GND	AH24	GND	F02
DS1_DQS(1)	C19	GND	AH28	GND	F06
DS1_DQS(2)	D19	GND	AH32	GND	F10
DS1_DQS(3)	R18	GND	AK04	GND	F14

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position
GND	F20
GND	F24
GND	F28
GND	F32
GND	H04
GND	H08
GND	H12
GND	H16
GND	H18
GND	H22
GND	H26
GND	H30
GND	K02
GND	K06
GND	K10
GND	K14
GND	K20
GND	K24
GND	K28
GND	K32
GND	M04
GND	M08
GND	M12
GND	M16
GND	M18
GND	M22
GND	M26
GND	M30
GND	P02
GND	P06
GND	P10
GND	P14
GND	P17
GND	P20
GND	P24
GND	P28
GND	P32
GND	R15
GND	R19
GND	T04
GND	T08
GND	T12
GND	T16
GND	T18
GND	T22
GND	T26
GND	T30
GND	U14
GND	U17
GND	U20
GND	V04
GND	V08
GND	V12
GND	V16
GND	V18
GND	V22
GND	V26
GND	V30
GND	W15
GND	W19
GND	Y02
GND	Y06
GND	Y10
GND	Y14
GND	Y17
GND	Y20
GND	Y24
GND	Y28
GND	Y32
I_FreeQ_Th	V21
$\overline{\text{IEW\_Clk\_A}}$	AL05
IEW_Clk_A	AN05
$\overline{\text{IEW\_Clk\_B}}$	A05
IEW_Clk_B	C05
JTAG_TCk	AA22
JTAG_TDI	W22
JTAG_TDO	AA23
JTAG_TMS	U22
$\overline{\text{JTAG\_TRst}}$	T25
LU_Addr(00)	AA09
LU_Addr(01)	Y11
LU_Addr(02)	AA10
LU_Addr(03)	AB07
LU_Addr(04)	AC09
LU_Addr(05)	AE06
LU_Addr(06)	AE07
LU_Addr(07)	AC01
LU_Addr(08)	R04
LU_Addr(09)	AG05
LU_Addr(10)	AG06
LU_Addr(11)	AC04
LU_Addr(12)	AD07
LU_Addr(13)	AF07
LU_Addr(14)	AB05
LU_Addr(15)	AE08
LU_Addr(16)	AB09
LU_Addr(17)	AA05
LU_Addr(18)	AA11
LU_Addr(19)	U07
LU_Clk	AJ03
LU_Data(00)	U15
LU_Data(01)	U13
LU_Data(02)	T09
LU_Data(03)	T07
LU_Data(04)	R07

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
LU_Data(05)	R08	MGrant_A(0)	V19	PCI_AD(30)	AK33
LU_Data(06)	W06	MGrant_A(1)	U19	PCI_AD(31)	AK31
LU_Data(07)	W05	MGrant_B(0)	AG27	PCI_Bus_M_Int	AC23
LU_Data(08)	U08	MGrant_B(1)	AE25	PCI_Bus_NM_Int	G27
LU_Data(09)	V07	$\overline{\text{Operational}}$	C32	$\overline{\text{PCI\_CBE(0)}}$	AC28
LU_Data(10)	V09	PCI_AD(00)	AB29	$\overline{\text{PCI\_CBE(1)}}$	AD25
LU_Data(11)	U12	PCI_AD(01)	AB27	$\overline{\text{PCI\_CBE(2)}}$	AF31
LU_Data(12)	V15	PCI_AD(02)	AB25	$\overline{\text{PCI\_CBE(3)}}$	AH31
LU_Data(13)	W04	PCI_AD(03)	AC33	PCI_Clk	AF33
LU_Data(14)	V11	PCI_AD(04)	AC32	$\overline{\text{PCI\_DevSel}}$	AE29
LU_Data(15)	W07	PCI_AD(05)	AC31	$\overline{\text{PCI\_Frame}}$	AE26
LU_Data(16)	W08	PCI_AD(06)	AC30	$\overline{\text{PCI\_Grant}}$	AL32
LU_Data(17)	Y07	PCI_AD(07)	AC29	PCI_IDSel	AH33
LU_Data(18)	V13	PCI_AD(08)	AC27	$\overline{\text{PCI\_IntA}}$	AM33
LU_Data(19)	W13	PCI_AD(09)	AC26	$\overline{\text{PCI\_IRdy}}$	AE27
LU_Data(20)	W01	PCI_AD(10)	AC25	PCI_Par	AE33
LU_Data(21)	W09	PCI_AD(11)	AC24	$\overline{\text{PCI\_PErr}}$	AE31
LU_Data(22)	Y09	PCI_AD(12)	AD33	$\overline{\text{PCI\_Request}}$	AL33
LU_Data(23)	AA06	PCI_AD(13)	AD31	$\overline{\text{PCI\_SErr}}$	AE32
LU_Data(24)	T15	PCI_AD(14)	AD29	PCI_Speed	M07
LU_Data(25)	W10	PCI_AD(15)	AD27	$\overline{\text{PCI\_Stop}}$	AE30
LU_Data(26)	W12	PCI_AD(16)	AF29	$\overline{\text{PCI\_TRdy}}$	AE28
LU_Data(27)	W14	PCI_AD(17)	AF27	PGM_GND	J09
LU_Data(28)	AA07	PCI_AD(18)	AG33	PGM_VDD	L11
LU_Data(29)	AA08	PCI_AD(19)	AG32	PIO(0)	N09
LU_Data(30)	U01	PCI_AD(20)	AG31	PIO(1)	N08
LU_Data(31)	W11	PCI_AD(21)	AG30	PIO(2)	N06
LU_Data(32)	Y05	PCI_AD(22)	AG29	PLLA_GND	AG01
LU_Data(33)	R05	PCI_AD(23)	AG28	PLLA_VDD	AJ01
LU_Data(34)	U10	PCI_AD(24)	AH29	PLLB_GND	G01
LU_Data(35)	U03	PCI_AD(25)	AJ33	PLLB_V_{DD}	E01
LU_R_$\overline{\text{Wrt}}$	AC08	PCI_AD(26)	AJ32	PLLC_GND	G33
MG_Clk	J07	PCI_AD(27)	AJ31	PLLC_V_{DD}	E33
MG_Data	J06	PCI_AD(28)	AJ30	RES_Data	T21
MG_nIntr	J08	PCI_AD(29)	AJ29	RES_Sync	U21

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
Reserved_IO(17)	AJ14	Reserved_IO(53)	P21	SCH_Data(08)	A06
Reserved_IO(18)	AN16	Reserved_IO(54)	R21	SCH_Data(09)	E08
Reserved_IO(19)	P27	Reserved_IO(55)	U23	SCH_Data(10)	G06
Reserved_IO(20)	P25	Reserved_IO(56)	V23	SCH_Data(11)	G07
Reserved_IO(21)	P23	Reserved_IO(57)	AA24	SCH_Data(12)	C06
Reserved_IO(22)	R33	Reserved_IO(58)	N22	SCH_Data(13)	D07
Reserved_IO(23)	R32	$\overline{\text{Reset}}$	E29	SCH_Data(14)	C09
Reserved_IO(24)	R31	SCH_Addr(00)	A07	SCH_Data(15)	A08
Reserved_IO(25)	R30	SCH_Addr(01)	B05	SCH_Data(16)	J10
Reserved_IO(26)	R29	SCH_Addr(02)	A04	SCH_Data(17)	G08
Reserved_IO(28)	R27	SCH_Addr(03)	E06	SCH_R_$\overline{\text{Wrt}}$	G05
Reserved_IO(29)	R26	SCH_Addr(04)	E07	Send_Grant_A	W20
Reserved_IO(30)	R25	SCH_Addr(05)	E04	Send_Grant_B	T19
Reserved_IO(31)	R24	SCH_Addr(06)	H07	Spare_Tst_Rcvr(0)	U05
Reserved_IO(32)	R23	SCH_Addr(07)	F05	Spare_Tst_Rcvr(1)	E03
Reserved_IO(33)	T33	SCH_Addr(08)	F07	Spare_Tst_Rcvr(2)	A03
Reserved_IO(34)	T31	SCH_Addr(09)	C03	Spare_Tst_Rcvr(3)	T01
Reserved_IO(35)	T29	SCH_Addr(10)	D05	Spare_Tst_Rcvr(4)	AL01
Reserved_IO(36)	T27	SCH_Addr(11)	A02	Spare_Tst_Rcvr(5)	G03
Reserved_IO(37)	R22	SCH_Addr(12)	C04	Spare_Tst_Rcvr(6)	V01
Reserved_IO(38)	T23	SCH_Addr(13)	B03	Spare_Tst_Rcvr(7)	V03
Reserved_IO(39)	V25	SCH_Addr(14)	C02	Spare_Tst_Rcvr(8)	T03
Reserved_IO(40)	U32	SCH_Addr(15)	D03	Spare_Tst_Rcvr(9)	U33
Reserved_IO(41)	U31	SCH_Addr(16)	B01	Testmode(0)	V05
Reserved_IO(42)	U30	SCH_Addr(17)	C01	Testmode(1)	U06
Reserved_IO(43)	U29	SCH_Addr(18)	E05	Thermal_In	U04
Reserved_IO(44)	U28	SCH_Clk	C07	Thermal_Out	U02
Reserved_IO(45)	U27	SCH_Data(00)	A10	Unused	AA15
Reserved_IO(46)	U26	SCH_Data(01)	C10	Unused	AB17
Reserved_IO(47)	U25	SCH_Data(02)	F09	Unused	AC16
Reserved_IO(48)	U24	SCH_Data(03)	J11	Unused	AD15
Reserved_IO(49)	V33	SCH_Data(04)	L12	Unused	AE12
Reserved_IO(50)	Y21	SCH_Data(05)	G09	Unused	AE14
Reserved_IO(51)	W21	SCH_Data(06)	B09	Unused	AF13
Reserved_IO(52)	R20	SCH_Data(07)	A09	Unused	AG12

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
Unused	AH11	Unused	R16	V_{DD}	D10
Unused	AH13	Unused	R28	V_{DD}	D20
Unused	AH15	Unused	U07	V_{DD}	D28
Unused	AJ10	Unused	U11	V_{DD}	D32
Unused	AJ11	Unused	W16	V_{DD}	F04
Unused	AJ13	Unused	Y15	V_{DD}	F08
Unused	AJ15	V_{DD}	AA13	V_{DD}	F16
Unused	AK09	V_{DD}	AA21	V_{DD}	F22
Unused	AK11	V_{DD}	AB02	V_{DD}	H02
Unused	AK13	V_{DD}	AB10	V_{DD}	H14
Unused	AL12	V_{DD}	AB28	V_{DD}	H24
Unused	AL13	V_{DD}	AD04	V_{DD}	H28
Unused	AL14	V_{DD}	AD16	V_{DD}	K08
Unused	AL16	V_{DD}	AD22	V_{DD}	K12
Unused	AM13	V_{DD}	AD26	V_{DD}	K18
Unused	AM15	V_{DD}	AF06	V_{DD}	K30
Unused	AN11	V_{DD}	AF10	V_{DD}	M06
Unused	AN12	V_{DD}	AF20	V_{DD}	M24
Unused	AN13	V_{DD}	AF32	V_{DD}	M32
Unused	AN14	V_{DD}	AH12	V_{DD}	N13
Unused	AN15	V_{DD}	AH18	V_{DD}	N21
Unused	K09	V_{DD}	AH26	V_{DD}	P04
Unused	L01	V_{DD}	AH30	V_{DD}	P16
Unused	L08	V_{DD}	AK02	V_{DD}	P18
Unused	L09	V_{DD}	AK06	V_{DD}	P26
Unused	M05	V_{DD}	AK14	V_{DD}	T02
Unused	M09	V_{DD}	AK24	V_{DD}	T10
Unused	M11	V_{DD}	AM08	V_{DD}	T14
Unused	N05	V_{DD}	AM16	V_{DD}	T17
Unused	N10	V_{DD}	AM22	V_{DD}	T20
Unused	N12	V_{DD}	AM30	V_{DD}	T28
Unused	P11	V_{DD}	B04	V_{DD}	U16
Unused	P13	V_{DD}	B12	V_{DD}	U18
Unused	R11	V_{DD}	B18	V_{DD}	V06
Unused	R12	V_{DD}	B26	V_{DD}	V14

Table 2-38. Complete Signal Pin Listing by Signal Name (Continued)

Signal Name	Grid Position	Signal Name	Grid Position	Signal Name	Grid Position
V_{DD}	V17	2.5 V	AF24	2.5 V	K22
V_{DD}	V20	2.5 V	AH04	2.5 V	M02
V_{DD}	V24	2.5 V	AH08	2.5 V	M10
V_{DD}	V32	2.5 V	AH16	2.5 V	M14
V_{DD}	Y08	2.5 V	AH22	2.5 V	M20
V_{DD}	Y16	2.5 V	AK10	2.5 V	P08
V_{DD}	Y18	2.5 V	AK20	2.5 V	P12
V_{DD}	Y30	2.5 V	AK28	2.5 V	T06
VRef1(0)	AD23	2.5 V	AM04	2.5 V	V02
VRef1(1)	K11	2.5 V	AM12	2.5 V	V10
VRef1(2)	AC10	2.5 V	AM18	2.5 V	Y04
VRef2(0)	AC19	2.5 V	AM26	2.5 V	Y12
VRef2(1)	AD17	2.5 V	B08	3.3 V	AB24
VRef2(2)	AC13	2.5 V	B16	3.3 V	AB32
VRef2(3)	L14	2.5 V	B22	3.3 V	AD30
VRef2(4)	K17	2.5 V	B30	3.3 V	AF28
VRef2(5)	L21	2.5 V	D02	3.3 V	AK32
VRef2(6)	Y13	2.5 V	D06	3.3 V	F30
VRef2(7)	N11	2.5 V	D14	3.3 V	H32
VRef2(8)	L10	2.5 V	D24	3.3 V	K26
2.5 V	AB06	2.5 V	F12	3.3 V	M28
2.5 V	AB14	2.5 V	F18	3.3 V	P22
2.5 V	AB20	2.5 V	F26	3.3 V	P30
2.5 V	AD08	2.5 V	H06	3.3 V	T24
2.5 V	AD12	2.5 V	H10	3.3 V	T32
2.5 V	AD18	2.5 V	H20	3.3 V	V28
2.5 V	AF02	2.5 V	K04	3.3 V	Y22
2.5 V	AF14	2.5 V	K16	3.3 V	Y26

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

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
AA05	LU_Addr(17)
AA06	LU_Data(23)
AA07	LU_Data(28)
AA08	LU_Data(29)
AA09	LU_Addr(00)
AA10	LU_Addr(02)
AA11	LU_Addr(18)
AA12	D0_Data(00)
AA13	V_{DD}
AA14	D0_Addr(04)
AA15	Unused
AA16	D3_Data(01)
AA17	D3_Addr(06)
AA18	D2_Data(03)
AA19	D2_Addr(12)
AA20	DA_BA(1)
AA21	V_{DD}
AA22	JTAG_TCk
AA23	JTAG_TDO
AA24	Reserved_IO(57)
AA25	DMU_A(28)
AA26	DMU_A(27)
AA27	DMU_A(26)
AA28	DMU_A(25)
AA29	DMU_A(24)
AA30	DMU_A(23)
AA31	DMU_A(22)
AA32	DMU_A(21)
AA33	DMU_A(20)
AB02	V_{DD}
AB04	GND
AB05	LU_Addr(14)
AB06	2.5 V
AB07	LU_Addr(03)
AB08	GND
AB09	LU_Addr(16)
AB10	V_{DD}
AB11	D0_Data(03)
AB12	GND
AB13	D0_Data(28)
AB14	2.5 V
AB15	D0_Addr(05)
AB16	GND
AB17	Unused
AB18	GND
AB19	D2_DQS(0)
AB20	2.5 V
AB21	D6_Data(04)
AB22	GND
AB23	$\overline{\text{C405\_Debug\_Halt}}$
AB24	3.3 V
AB25	PCI_AD(02)
AB26	GND
AB27	PCI_AD(01)
AB28	V_{DD}
AB29	PCI_AD(00)
AB30	GND
AB31	DMU_A(30)
AB32	3.3 V
AB33	DMU_A(29)
AC01	LU_Addr(07)
AC04	LU_Addr(11)
AC08	LU_R_$\overline{\text{Wrt}}$
AC09	LU_Addr(04)
AC10	VRef1(2)
AC11	D0_Data(13)
AC12	D0_DQS(1)
AC13	VRef2(2)
AC14	D0_Addr(10)
AC15	D0_Data(29)
AC16	Unused
AC17	D3_DQS(1)
AC18	D3_Addr(07)
AC19	VRef2(0)
AC20	D2_Addr(05)
AC21	D6_Data(10)
AC22	D6_Data(15)
AC23	PCI_Bus_M_Int
AC24	PCI_AD(11)
AC25	PCI_AD(10)
AC26	PCI_AD(09)
AC27	PCI_AD(08)
AC28	$\overline{\text{PCI\_CBE(0)}}$
AC29	PCI_AD(07)
AC30	PCI_AD(06)
AC31	PCI_AD(05)
AC32	PCI_AD(04)
AC33	PCI_AD(03)
AD02	GND
AD04	V_{DD}
AD06	GND
AD07	LU_Addr(12)
AD08	2.5 V
AD09	D0_Data(01)
AD10	GND
AD11	D0_Data(23)
AD12	2.5 V
AD13	DB_BA(1)
AD14	GND
AD15	Unused
AD16	V_{DD}
AD17	VRef2(1)
AD18	2.5 V
AD19	$\overline{\text{D3\_WE}}$
AD20	GND

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
AD21	D2_Addr(06)
AD22	V_{DD}
AD23	VRef1(0)
AD24	GND
AD25	$\overline{\text{PCI\_CBE(1)}}$
AD26	V_{DD}
AD27	PCI_AD(15)
AD28	GND
AD29	PCI_AD(14)
AD30	3.3 V
AD31	PCI_AD(13)
AD32	GND
AD33	PCI_AD(12)
AE06	LU_Addr(05)
AE07	LU_Addr(06)
AE08	LU_Addr(15)
AE09	D0_Data(11)
AE10	D0_Data(22)
AE11	D0_DQS(2)
AE12	Unused
AE13	$\overline{\text{DB\_RAS}}$
AE14	Unused
AE15	D3_Data(03)
AE16	D3_Data(09)
AE17	D2_Data(08)
AE18	D3_Addr(05)
AE19	D3_Addr(12)
AE20	D2_Data(07)
AE21	D2_Data(09)
AE22	D6_Data(14)
AE23	D6_Data(09)
AE24	D6_Addr(02)
AE25	MGrant_B(1)
AE26	$\overline{\text{PCI\_Frame}}$
AE27	$\overline{\text{PCI\_IRdy}}$
AE28	$\overline{\text{PCI\_TRdy}}$
AE29	$\overline{\text{PCI\_DevSel}}$
AE30	$\overline{\text{PCI\_Stop}}$
AE31	$\overline{\text{PCI\_PErr}}$
AE32	$\overline{\text{PCI\_SErr}}$
AE33	PCI_Par
AF02	2.5 V
AF04	GND
AF06	V_{DD}
AF07	LU_Addr(13)
AF08	GND
AF09	D0_Data(20)
AF10	V_{DD}
AF11	D0_Addr(12)
AF12	GND
AF13	Unused
AF14	2.5 V
AF15	D3_Data(14)
AF16	GND
AF17	D3_Addr(02)
AF18	GND
AF19	D3_Data(15)
AF20	V_{DD}
AF21	D6_DQS_Par(01)
AF22	GND
AF23	D6_ByteEn(1)
AF24	2.5 V
AF25	D6_Addr(04)
AF26	GND
AF27	PCI_AD(17)
AF28	3.3 V
AF29	PCI_AD(16)
AF30	GND
AF31	$\overline{\text{PCI\_CBE(2)}}$
AF32	V_{DD}
AF33	PCI_Clk
AG01	PLLA_GND
AG03	D0_Data(09)
AG05	LU_Addr(09)
AG06	LU_Addr(10)
AG07	D0_Data(02)
AG08	D0_Data(21)
AG09	D0_DQS(0)
AG10	D0_Addr(11)
AG11	DB_BA(0)
AG12	Unused
AG13	D3_Data(00)
AG14	D3_Data(02)
AG15	D3_Data(13)
AG16	D3_Data(10)
AG17	D3_Data(12)
AG18	D3_Addr(04)
AG19	D3_Addr(00)
AG20	D3_DQS(0)
AG21	D6_Addr(11)
AG22	D2_Data(10)
AG23	D2_Addr(07)
AG24	D6_ByteEn(0)
AG25	$\overline{\text{DA\_RAS}}$
AG26	D6_Addr(03)
AG27	MGrant_B(0)
AG28	PCI_AD(23)
AG29	PCI_AD(22)
AG30	PCI_AD(21)
AG31	PCI_AD(20)
AG32	PCI_AD(19)
AG33	PCI_AD(18)
AH02	GND
AH04	2.5 V
AH05	DE_BA(1)

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
AH06	GND	AJ09	D0_Addr(02)	AK12	GND
AH07	D0_Data(10)	AJ10	Unused	AK13	Unused
AH08	2.5 V	AJ11	Unused	AK14	V_{DD}
AH09	D0_DQS(3)	AJ12	$\overline{\text{DB\_CAS}}$	AK15	D3_Data(07)
AH10	GND	AJ13	Unused	AK16	GND
AH11	Unused	AJ14	Reserved_IO(17)	AK17	D3_Addr(11)
AH12	V_{DD}	AJ15	Unused	AK18	GND
AH13	Unused	AJ16	D3_Addr(01)	AK19	D3_Addr(08)
AH14	GND	AJ17	D3_Addr(03)	AK20	2.5 V
AH15	Unused	AJ18	$\overline{\text{DB\_Clk}}$	AK21	D2_Data(13)
AH16	2.5 V	AJ19	D2_Data(01)	AK22	GND
AH17	D3_Data(11)	AJ20	D2_Data(04)	AK23	D2_Addr(10)
AH18	V_{DD}	AJ21	D2_Data(14)	AK24	V_{DD}
AH19	DB_Clk	AJ22	D2_Addr(00)	AK25	D2_DQS(1)
AH20	GND	AJ23	D2_Addr(11)	AK26	GND
AH21	D2_Addr(01)	AJ24	$\overline{\text{D2\_WE}}$	AK27	D6_Data(13)
AH22	2.5 V	AJ25	$\overline{\text{D6\_WE}}$	AK28	2.5 V
AH23	D2_Addr(04)	AJ26	D6_Data(12)	AK29	D6_Addr(08)
AH24	GND	AJ27	D6_Addr(07)	AK30	GND
AH25	D6_Data(08)	AJ28	D6_Addr(09)	AK31	PCI_AD(31)
AH26	V_{DD}	AJ29	PCI_AD(29)	AK32	3.3 V
AH27	D6_Addr(12)	AJ30	PCI_AD(28)	AK33	PCI_AD(30)
AH28	GND	AJ31	PCI_AD(27)	AL01	Spare_Tst_Rcvr(4)
AH29	PCI_AD(24)	AJ32	PCI_AD(26)	AL02	$\overline{\text{DE\_CAS}}$
AH30	V_{DD}	AJ33	PCI_AD(25)	AL03	D0_Data(12)
AH31	$\overline{\text{PCI\_CBE(3)}}$	AK02	V_{DD}	AL04	D0_Data(08)
AH32	GND	AK03	$\overline{\text{DE\_RAS}}$	AL05	$\overline{\text{IEW\_Clk\_A}}$
AH33	PCI_IDSel	AK04	GND	AL06	D0_Data(26)
AJ01	PLLA_VDD	AK05	D0_Data(15)	AL07	D0_Data(19)
AJ03	LU_Clk	AK06	V_{DD}	AL08	D0_Addr(07)
AJ04	DE_Clk	AK07	D0_Data(30)	AL09	D0_Data(25)
AJ05	DE_$\overline{\text{Clk}}$	AK08	GND	AL10	D0_Addr(00)
AJ06	D0_Data(14)	AK09	Unused	AL11	D0_Addr(09)
AJ07	D0_Data(16)	AK10	2.5 V	AL12	Unused
AJ08	D0_Data(31)	AK11	Unused	AL13	Unused

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
AL14	Unused
AL15	D3_Data(06)
AL16	Unused
AL17	D3_Data(04)
AL18	$\overline{\text{D3\_CS}}$
AL19	D3_Addr(09)
AL20	D2_Data(05)
AL21	D2_Addr(09)
AL22	$\overline{\text{D2\_CS}}$
AL23	D6_Data(00)
AL24	D6_Data(07)
AL25	DA_$\overline{\text{Clk}}$
AL26	D6_Data(02)
AL27	D6_Addr(05)
AL28	D6_Addr(00)
AL29	D6_Addr(10)
AL30	D6_DQS(0)
AL31	$\overline{\text{D6\_CS}}$
AL32	$\overline{\text{PCI\_Grant}}$
AL33	$\overline{\text{PCI\_Request}}$
AM01	DE_BA(0)
AM02	GND
AM03	D0_Data(05)
AM04	2.5 V
AM05	D0_Data(27)
AM06	GND
AM07	D0_Addr(06)
AM08	V_{DD}
AM09	$\overline{\text{D0\_WE}}$
AM10	GND
AM11	D0_Addr(08)
AM12	2.5 V
AM13	Unused
AM14	GND
AM15	Unused
AM16	V_{DD}
AM17	D3_Data(05)
AM18	2.5 V
AM19	D2_Data(12)
AM20	GND
AM21	D2_Addr(03)
AM22	V_{DD}
AM23	D6_Data(01)
AM24	GND
AM25	DA_Clk
AM26	2.5 V
AM27	D6_Data(03)
AM28	GND
AM29	D6_Parity(00)
AM30	V_{DD}
AM31	D6_DQS(3)
AM32	GND
AM33	$\overline{\text{PCI\_IntA}}$
AN01	D0_Data(06)
AN02	D0_Data(04)
AN03	D0_Data(07)
AN04	D0_Data(17)
AN05	IEW_Clk_A
AN06	D0_Addr(03)
AN07	D0_Data(18)
AN08	D0_Data(24)
AN09	$\overline{\text{D0\_CS}}$
AN10	D0_Addr(01)
AN11	Unused
AN12	Unused
AN13	Unused
AN14	Unused
AN15	Unused
AN16	Reserved_IO(18)
AN17	D3_Addr(10)
AN18	D2_Data(00)
AN19	D2_Data(06)
AN20	D2_Data(11)
AN21	D2_Addr(02)
AN22	D2_Addr(08)
AN23	D6_Parity(01)
AN24	D6_Data(06)
AN25	D6_Data(11)
AN26	D6_Addr(01)
AN27	$\overline{\text{DA\_CAS}}$
AN28	D6_Data(05)
AN29	DA_BA(0)
AN30	D6_Addr(06)
AN31	D6_DQS(1)
AN32	D6_DQS_Par(00)
AN33	D6_DQS(2)
A01	DUM
A02	SCH_Addr(11)
A03	Spare_Tst_Rcvr(2)
A04	SCH_Addr(02)
A05	$\overline{\text{IEW\_Clk\_B}}$
A06	SCH_Data(08)
A07	SCH_Addr(00)
A08	SCH_Data(15)
A09	SCH_Data(07)
A10	SCH_Data(00)
A11	D4_Addr(01)
A12	DD_BA(0)
A13	D4_Data(31)
A14	D4_Data(24)
A15	D4_Data(19)
A16	D4_Data(11)
A17	D4_Data(04)
A18	DS1_Data(26)
A19	DS1_Data(19)

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name	Grid Position	Signal Name	Grid Position	Signal Name
A20	DS1_Data(14)	B22	2.5 V	C24	DS0_DQS(1)
A21	DS1_Data(06)	B23	DS0_Addr(05)	C25	DS0_Data(21)
A22	DS1_Data(00)	B24	GND	C26	DS0_Addr(04)
A23	DS0_Addr(10)	B25	DS0_Data(29)	C27	DS0_Data(15)
A24	DS0_DQS(2)	B26	V_{DD}	C28	DS0_Data(22)
A25	DS0_Data(28)	B27	DS0_Addr(03)	C29	DS0_Data(08)
A26	DS0_Data(20)	B28	GND	C30	DS0_Data(04)
A27	DS0_Data(14)	B29	DS0_Data(23)	C31	DS0_Data(05)
A28	DS0_Addr(00)	B30	2.5 V	C32	$\overline{\text{Operational}}$
A29	DS0_Data(09)	B31	DS0_Data(02)	C33	Core_Clock
A30	DS0_Data(13)	B32	GND	D02	2.5 V
A31	$\overline{\text{DS0\_CS}}$	B33	Clock125	D03	SCH_Addr(15)
A32	DS0_Data(01)	C01	SCH_Addr(17)	D04	GND
A33	DS0_Data(03)	C02	SCH_Addr(14)	D05	SCH_Addr(10)
B01	SCH_Addr(16)	C03	SCH_Addr(09)	D06	2.5 V
B02	GND	C04	SCH_Addr(12)	D07	SCH_Data(13)
B03	SCH_Addr(13)	C05	IEW_Clk_B	D08	GND
B04	V_{DD}	C06	SCH_Data(12)	D09	D4_Addr(08)
B05	SCH_Addr(01)	C07	SCH_Clk	D10	V_{DD}
B06	GND	C08	D4_Addr(09)	D11	D4_DQS(0)
B07	D4_Addr(12)	C09	SCH_Data(14)	D12	GND
B08	2.5 V	C10	SCH_Data(01)	D13	D4_Data(26)
B09	SCH_Data(06)	C11	D4_Addr(06)	D14	2.5 V
B10	GND	C12	D4_Addr(00)	D15	D4_Data(02)
B11	D4_Addr(07)	C13	DD_Clk	D16	GND
B12	V_{DD}	C14	D4_Data(17)	D17	D4_Data(05)
B13	DD_$\overline{\text{Clk}}$	C15	D4_Data(03)	D18	GND
B14	GND	C16	D4_Data(10)	D19	DS1_DQS(2)
B15	D4_Data(25)	C17	$\overline{\text{DS1\_WE}}$	D20	V_{DD}
B16	2.5 V	C18	DS1_Data(27)	D21	DS1_Data(12)
B17	DS1_DQS(0)	C19	DS1_DQS(1)	D22	GND
B18	V_{DD}	C20	DS1_Data(21)	D23	DC_$\overline{\text{Clk}}$
B19	DS1_Data(13)	C21	$\overline{\text{DC\_RAS}}$	D24	2.5 V
B20	GND	C22	DS0_Addr(11)	D25	DC_BA(0)
B21	DS1_Data(05)	C23	DS0_Addr(06)	D26	GND

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
D27	DS0_Data(26)
D28	V_{DD}
D29	DS0_Data(11)
D30	GND
D31	DMU_D(01)
D32	V_{DD}
D33	DMU_D(00)
E01	PLLB_V_{DD}
E03	Spare_Tst_Rcvr(1)
E04	SCH_Addr(05)
E05	SCH_Addr(18)
E06	SCH_Addr(03)
E07	SCH_Addr(04)
E08	SCH_Data(09)
E09	D4_Data(18)
E10	$\overline{\text{D4\_CS}}$
E11	D4_DQS(1)
E12	D4_Data(29)
E13	D4_Data(27)
E14	D4_Data(16)
E15	D4_Data(12)
E16	$\overline{\text{DS1\_CS}}$
E17	DS1_Addr(03)
E18	DS1_Data(20)
E19	DS1_Data(25)
E20	DS1_Data(22)
E21	DS1_Data(11)
E22	DS1_Data(08)
E23	DC_Clk
E24	DS0_Addr(12)
E25	DS0_DQS(3)
E26	DS0_Data(27)
E27	DS0_Data(12)
E28	DS0_Data(10)
E29	$\overline{\text{Reset}}$
E30	DMU_D(04)
E31	DMU_D(29)
E32	DMU_D(12)
E33	PLLC_V_{DD}
F02	GND
F04	V_{DD}
F05	SCH_Addr(07)
F06	GND
F07	SCH_Addr(08)
F08	V_{DD}
F09	SCH_Data(02)
F10	GND
F11	$\overline{\text{D4\_WE}}$
F12	2.5 V
F13	D4_Data(30)
F14	GND
F15	D4_Data(13)
F16	V_{DD}
F17	DS1_Addr(10)
F18	2.5 V
F19	DS1_Data(24)
F20	GND
F21	DS1_Data(07)
F22	V_{DD}
F23	DS1_Data(04)
F24	GND
F25	DS0_DQS(0)
F26	2.5 V
F27	DS0_Data(06)
F28	GND
F29	DMU_D(07)
F30	3.3 V
F31	DMU_D(06)
F32	GND
F33	DMU_D(05)
G01	PLLB_GND
G03	Spare_Tst_Rcvr(5)
G05	SCH_R_$\overline{\text{Wrt}}$
G06	SCH_Data(10)
G07	SCH_Data(11)
G08	SCH_Data(17)
G09	SCH_Data(05)
G10	D4_Addr(04)
G11	$\overline{\text{DD\_CAS}}$
G12	D4_Data(22)
G13	D4_Data(08)
G14	D4_Data(07)
G15	DS1_Addr(08)
G16	DS1_Addr(11)
G17	DS1_Addr(09)
G18	DS1_Addr(02)
G19	DS1_Addr(05)
G20	DS1_Data(30)
G21	DS1_Data(29)
G22	DS1_Data(15)
G23	DS1_Data(01)
G24	DS0_Addr(07)
G25	DS0_Data(30)
G26	DS0_Data(17)
G27	PCI_Bus_NM_Int
G28	DMU_D(02)
G29	DMU_D(11)
G30	DMU_D(10)
G31	DMU_D(30)
G32	DMU_D(08)
G33	PLLC_GND
H02	V_{DD}
H04	GND
H06	2.5 V
H07	SCH_Addr(06)

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
H08	GND
H09	D4_Data(06)
H10	2.5 V
H11	D4_Addr(03)
H12	GND
H13	D4_Data(23)
H14	V_{DD}
H15	DS1_Addr(07)
H16	GND
H17	DS1_Addr(04)
H18	GND
H19	DS1_Addr(06)
H20	2.5 V
H21	DS0_Data(07)
H22	GND
H23	DS0_Addr(08)
H24	V_{DD}
H25	DS0_Data(16)
H26	GND
H27	DMU_D(16)
H28	V_{DD}
H29	DMU_D(15)
H30	GND
H31	DMU_D(14)
H32	3.3 V
H33	DMU_D(13)
J06	MG_Data
J07	MG_Clk
J08	MG_nIntr
J09	PGM_GND
J10	SCH_Data(16)
J11	SCH_Data(03)
J12	D4_Addr(02)
J13	DD_BA(1)
J14	D4_Data(21)
J15	DS1_Data(17)
J16	DS1_Addr(12)
J17	DC_BA(1)
J18	DS1_Addr(01)
J19	DS1_Data(31)
J20	DS1_Data(18)
J21	DS1_Data(16)
J22	DS0_Addr(09)
J23	$\overline{\text{DS0\_WE}}$
J24	DS0_Data(18)
J25	DMU_D(25)
J26	DMU_D(24)
J27	DMU_D(23)
J28	DMU_D(22)
J29	DMU_D(03)
J30	DMU_D(20)
J31	DMU_D(19)
J32	DMU_D(18)
J33	DMU_D(17)
K02	GND
K04	2.5 V
K06	GND
K07	Boot_Picocode
K08	V_{DD}
K09	Unused
K10	GND
K11	VRef1(1)
K12	V_{DD}
K13	$\overline{\text{DD\_RAS}}$
K14	GND
K15	D4_Data(09)
K16	2.5 V
K17	VRef2(4)
K18	V_{DD}
K19	DS1_Data(28)
K20	GND
K21	DS1_Data(02)
K22	2.5 V
K23	DS0_Data(19)
K24	GND
K25	DMU_D(09)
K26	3.3 V
K27	DMU_D(21)
K28	GND
K29	DMU_D(28)
K30	V_{DD}
K31	DMU_D(27)
K32	GND
K33	DMU_D(26)
L01	Unused
L04	Boot_PPC
L08	Unused
L09	Unused
L10	VRef2(8)
L11	PGM_VDD
L12	SCH_Data(04)
L13	D4_Addr(05)
L14	VRef2(3)
L15	D4_Data(20)
L16	D4_Data(01)
L17	DS1_Data(10)
L18	DS1_Data(09)
L19	DS0_Data(25)
L20	DS1_Data(03)
L21	VRef2(5)
L22	DS0_Data(31)
L23	DMU_C(10)
L24	DMU_C(09)
L25	DMU_C(08)
L26	DMU_C(07)

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
L27	DMU_C(06)
L28	DMU_C(05)
L29	DMU_C(04)
L30	DMU_C(03)
L31	DMU_C(02)
L32	DMU_C(01)
L33	DMU_C(00)
M02	2.5 V
M04	GND
M05	Unused
M06	V_{DD}
M07	PCI_Speed
M08	GND
M09	Unused
M10	2.5 V
M11	Unused
M12	GND
M13	D4_Addr(10)
M14	2.5 V
M15	D4_DQS(3)
M16	GND
M17	D4_Data(00)
M18	GND
M19	DS0_Addr(02)
M20	2.5 V
M21	DS0_Data(24)
M22	GND
M23	DMU_C(16)
M24	V_{DD}
M25	DMU_C(15)
M26	GND
M27	DMU_C(14)
M28	3.3 V
M29	DMU_C(13)
M30	GND
M31	DMU_C(12)
M32	V_{DD}
M33	DMU_C(11)
N05	Unused
N06	PIO(2)
N07	cam_cp_response(13)
N08	PIO(1)
N09	PIO(0)
N10	Unused
N11	VRef2(7)
N12	Unused
N13	V_{DD}
N14	D4_Addr(11)
N15	D4_DQS(2)
N16	D4_Data(15)
N17	DS1_Addr(00)
N18	DS1_Data(23)
N19	$\overline{\text{DC\_CAS}}$
N20	DS0_Addr(01)
N21	V_{DD}
N22	Reserved_IO(58)
N23	DMU_C(27)
N24	DMU_C(26)
N25	DMU_C(25)
N26	DMU_C(24)
N27	DMU_C(23)
N28	DMU_C(22)
N29	DMU_C(21)
N30	DMU_C(20)
N31	DMU_C(19)
N32	DMU_C(18)
N33	DMU_C(17)
P02	GND
P04	V_{DD}
P05	cam_cp_response(02)
P06	GND
P07	cam_cp_response(05)
P08	2.5 V
P09	cam_cp_response(09)
P10	GND
P11	Unused
P12	2.5 V
P13	Unused
P14	GND
P15	D4_Data(28)
P16	V_{DD}
P17	GND
P18	V_{DD}
P19	DS0_Data(00)
P20	GND
P21	Reserved_IO(53)
P22	3.3 V
P23	Reserved_IO(21)
P24	GND
P25	Reserved_IO(20)
P26	V_{DD}
P27	Reserved_IO(19)
P28	GND
P29	DMU_C(30)
P30	3.3 V
P31	DMU_C(29)
P32	GND
P33	DMU_C(28)
R01	cam_cp_response(07)
R04	LU_Addr(08)
R05	LU_Data(33)
R06	cam_cp_response(03)
R07	LU_Data(04)
R08	LU_Data(05)
R09	cam_cp_response(10)

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

Grid Position	Signal Name
R10	cam_cp_response(08)
R11	Unused
R12	Unused
R13	cam_cp_response(12)
R14	cam_cp_response(11)
R15	GND
R16	Unused
R17	D4_Data(14)
R18	DS1_DQS(3)
R19	GND
R20	Reserved_IO(52)
R21	Reserved_IO(54)
R22	Reserved_IO(37)
R23	Reserved_IO(32)
R24	Reserved_IO(31)
R25	Reserved_IO(30)
R26	Reserved_IO(29)
R27	Reserved_IO(28)
R28	Unused
R29	Reserved_IO(26)
R30	Reserved_IO(25)
R31	Reserved_IO(24)
R32	Reserved_IO(23)
R33	Reserved_IO(22)
T01	Spare_Tst_Rcvr(3)
T02	V_{DD}
T03	Spare_Tst_Rcvr(8)
T04	GND
T05	cam_cp_response(06)
T06	2.5 V
T07	LU_Data(03)
T08	GND
T09	LU_Data(02)
T10	V_{DD}
T11	cam_cp_response(01)
T12	GND
T13	cam_cp_response(04)
T14	V_{DD}
T15	LU_Data(24)
T16	GND
T17	V_{DD}
T18	GND
T19	Send_Grant_B
T20	V_{DD}
T21	RES_Data
T22	GND
T23	Reserved_IO(38)
T24	3.3 V
T25	$\overline{\text{JTAG\_TRst}}$
T26	GND
T27	Reserved_IO(36)
T28	V_{DD}
T29	Reserved_IO(35)
T30	GND
T31	Reserved_IO(34)
T32	3.3 V
T33	Reserved_IO(33)
U01	LU_Data(30)
U02	Thermal_Out
U03	LU_Data(35)
U04	Thermal_In
U05	Spare_Tst_Rcvr(0)
U06	Testmode(1)
U07	LU_Addr(19)
U07	Unused
U08	LU_Data(08)
U09	cam_cp_response(00)
U10	LU_Data(34)
U11	Unused
U12	LU_Data(11)
U13	LU_Data(01)
U14	GND
U15	LU_Data(00)
U16	V_{DD}
U17	GND
U18	V_{DD}
U19	MGrant_A(1)
U20	GND
U21	RES_Sync
U22	JTAG_TMS
U23	Reserved_IO(55)
U24	Reserved_IO(48)
U25	Reserved_IO(47)
U26	Reserved_IO(46)
U27	Reserved_IO(45)
U28	Reserved_IO(44)
U29	Reserved_IO(43)
U30	Reserved_IO(42)
U31	Reserved_IO(41)
U32	Reserved_IO(40)
U33	Spare_Tst_Rcvr(9)
V01	Spare_Tst_Rcvr(6)
V02	2.5 V
V03	Spare_Tst_Rcvr(7)
V04	GND
V05	Testmode(0)
V06	V_{DD}
V07	LU_Data(09)
V08	GND
V09	LU_Data(10)
V10	2.5 V
V11	LU_Data(14)
V12	GND
V13	LU_Data(18)
V14	V_{DD}

Table 2-39. Complete Signal Pin Listing by Grid Position (Continued)

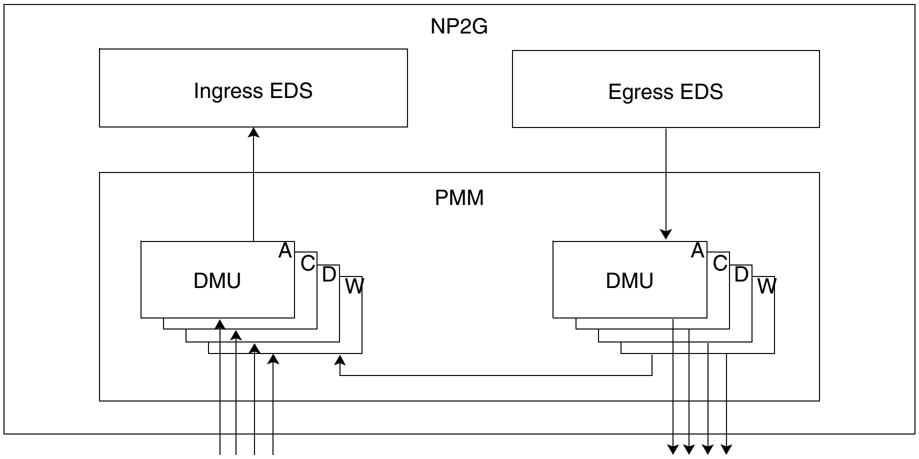
Grid Position	Signal Name
V15	LU_Data(12)
V16	GND
V17	V_{DD}
V18	GND
V19	MGrant_A(0)
V20	V_{DD}
V21	I_FreeQ_Th
V22	GND
V23	Reserved_IO(56)
V24	V_{DD}
V25	Reserved_IO(39)
V26	GND
V27	DMU_A(02)
V28	3.3 V
V29	DMU_A(01)
V30	GND
V31	DMU_A(00)
V32	V_{DD}
V33	Reserved_IO(49)
W01	LU_Data(20)
W04	LU_Data(13)
W05	LU_Data(07)
W06	LU_Data(06)
W07	LU_Data(15)
W08	LU_Data(16)
W09	LU_Data(21)
W10	LU_Data(25)
W11	LU_Data(31)
W12	LU_Data(26)
W13	LU_Data(19)
W14	LU_Data(27)
W15	GND
W16	Unused
W17	D3_Data(08)
W18	D2_Data(02)
W19	GND
W20	Send_Grant_A
W21	Reserved_IO(51)
W22	JTAG_TDI
W23	DMU_A(13)
W24	DMU_A(12)
W25	DMU_A(11)
W26	DMU_A(10)
W27	DMU_A(09)
W28	DMU_A(08)
W29	DMU_A(07)
W30	DMU_A(06)
W31	DMU_A(05)
W32	DMU_A(04)
W33	DMU_A(03)
Y02	GND
Y04	2.5 V
Y05	LU_Data(32)
Y06	GND
Y07	LU_Data(17)
Y08	V_{DD}
Y09	LU_Data(22)
Y10	GND
Y11	LU_Addr(01)
Y12	2.5 V
Y13	VRef2(6)
Y14	GND
Y15	Unused
Y16	V_{DD}
Y17	GND
Y18	V_{DD}
Y19	D2_Data(15)
Y20	GND
Y21	Reserved_IO(50)
Y22	3.3 V
Y23	DMU_A(19)
Y24	GND
Y25	DMU_A(18)
Y26	3.3 V
Y27	DMU_A(17)
Y28	GND
Y29	DMU_A(16)
Y30	V_{DD}
Y31	DMU_A(15)
Y32	GND
Y33	DMU_A(14)

3. Physical MAC Multiplexer

The Physical MAC Multiplexer (PMM) moves data between physical layer devices and the network processor. The PMM 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), as illustrated in *Figure 3-1*. Three DMUs (A, C, and D) can each be independently configured as an Ethernet Medium Access Control (MAC) or a packet over SONET (POS) MAC. The device keeps a complete set of performance statistics on a per-port basis in either mode. Each DMU moves data at 1 Gigabit per second (Gbps) in both the ingress and the egress directions. The Wrap DMU enables traffic generated by the NP2G's egress side to move to the ingress side of the device.

Figure 3-1. PMM Overview

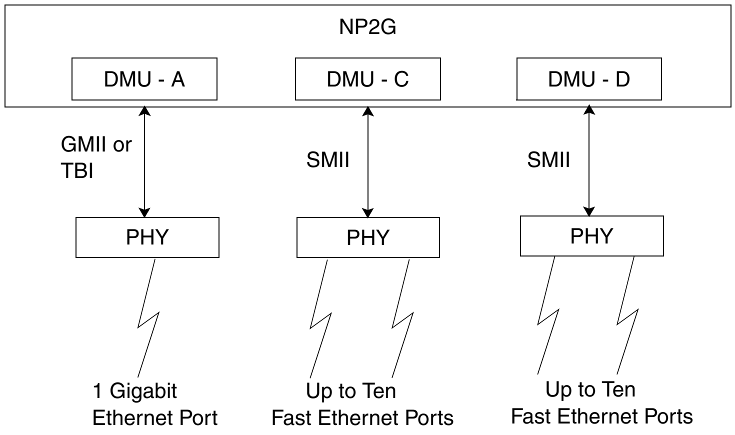


3.1 Ethernet Overview

For Ethernet operation, the uplink/control point port (DMU A) can be configured as either a Gigabit Media-Independent Interface (GMII) or a Ten-Bit Interface (TBI). The other data movers (DMU C & D) can be configured for either gigabit or fast ethernet. The same two gigabit modes are supported as on DMU A plus a Serial Media-Independent Interface (SMII). In this mode, the single DMU MAC can be configured as ten ports.

Figure 3-2 shows a sample NP2G with the PMM configured for Ethernet interfaces. DMU A is configured as a Gigabit Ethernet MAC. DMUs C and D are configured as Fast Ethernet MACs.

Figure 3-2. Ethernet Mode



3.1.1 Ethernet Interface Timing Diagrams

The following figures show timing diagrams for the Ethernet interfaces:

- *Figure 3-3*
- *Figure 3-4: GMII Timing Diagram* on page 111
- *Figure 3-5: TBI Timing Diagram* on page 111
- *Figure 3-6: GMII POS Mode Timing Diagram* on page 112.

Figure 3-3. SMII Timing Diagram

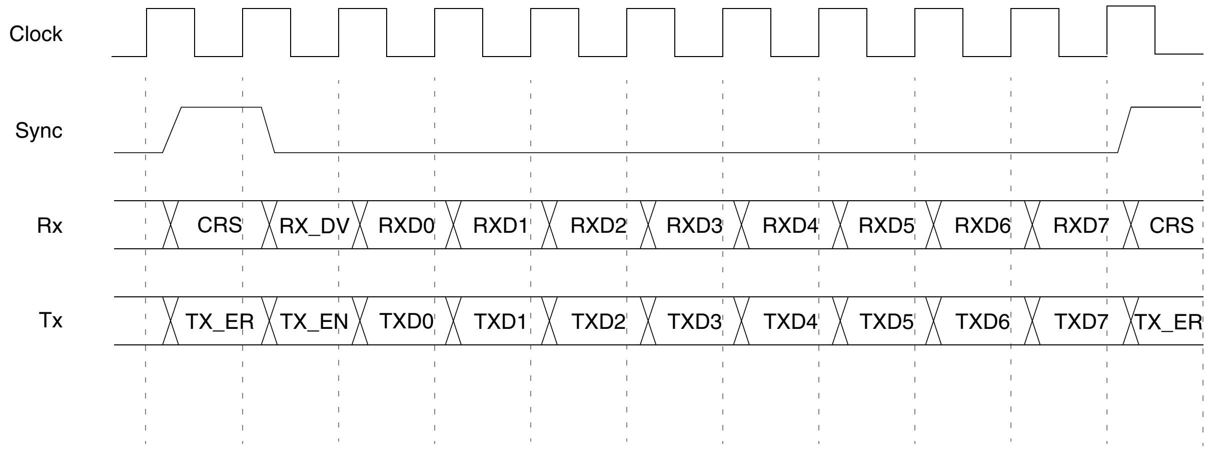


Figure 3-4. GMII Timing Diagram

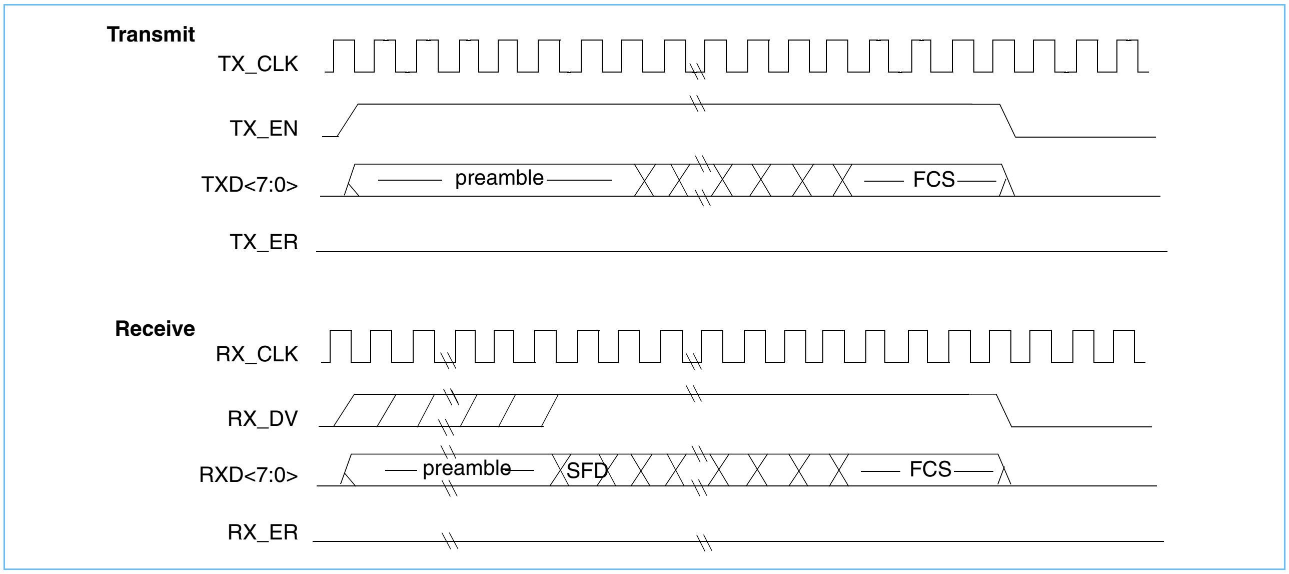


Figure 3-5. TBI Timing Diagram

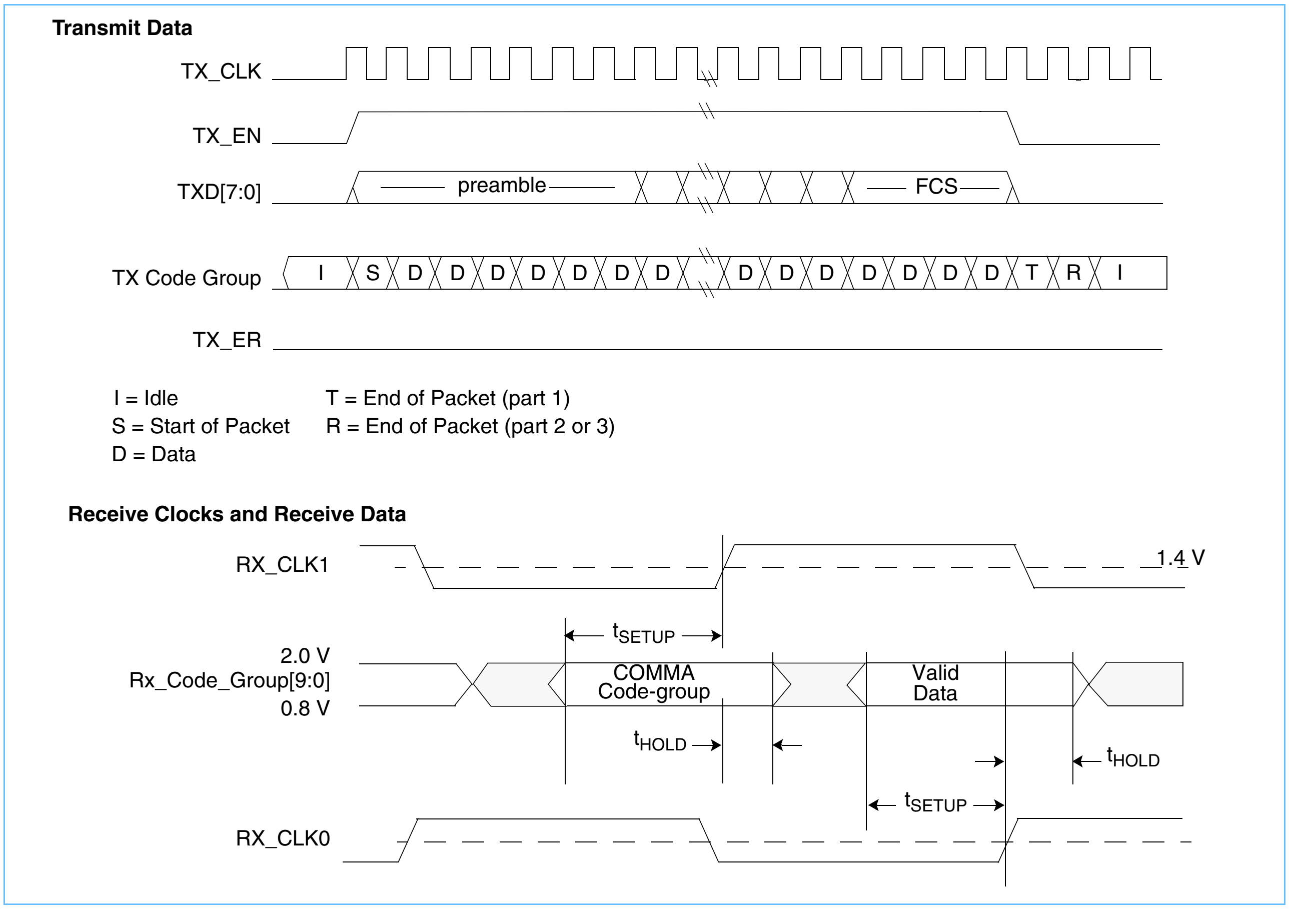
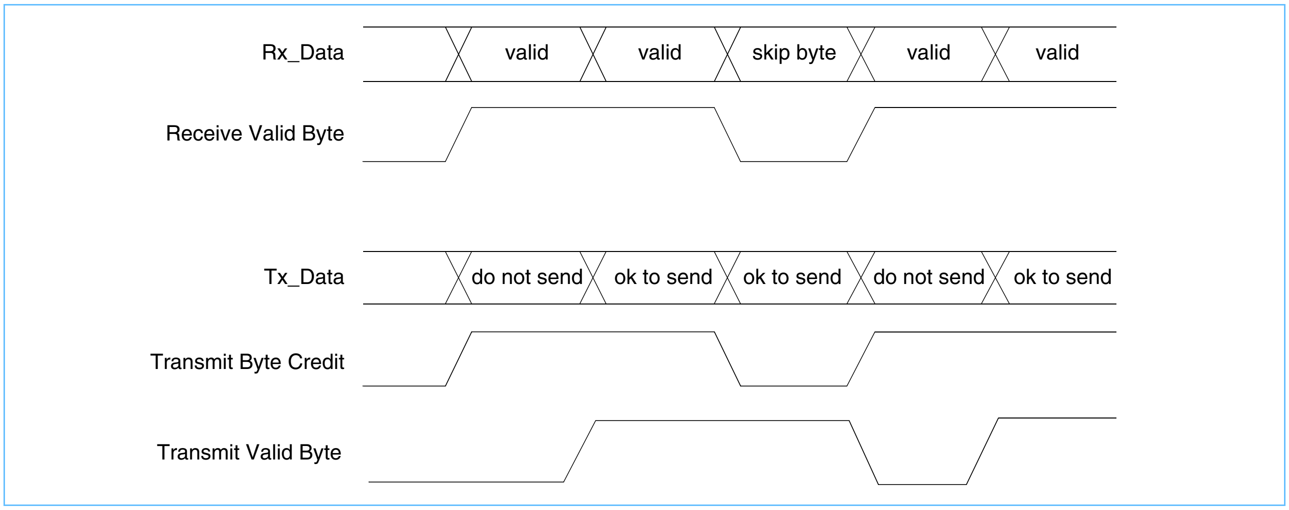


Figure 3-6. GMII POS Mode Timing Diagram



3.1.2 Ethernet Counters

The NP2G provides 36 Ethernet statistics counters per MAC (up to one million software-defined, hardware-assisted counters), enabling support of many standard Management Information Bases (MIBs) at wire speed.

Table 3-1: Ingress Ethernet Counters on page 112 and *Table 3-2: Egress Ethernet Counters* on page 114 show the statistics counters kept in each DMU when it operates in Ethernet mode. These counters are accessible through the Control Access Bus (CAB) interface.

Table 3-1. Ingress Ethernet Counters (Page 1 of 3)

Name / Group	Counter No.	Group	Description	Notes
Short Frames	x'00'		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

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. Further reception at the MAC is inhibited until the next frame starts.
4. The maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

Table 3-1. Ingress Ethernet Counters (Page 2 of 3)

Name / Group	Counter No.	Group	Description	Notes
Frames Received (512 to 1023 octets)	x‘06’	4	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 octets and the configured maximum length of a jumbo frame, or between 1519 octets and configured maximum length of a jumbo frame plus 4 if VLAN is asserted.	4
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 the configured maximum length of a jumbo frame in octets, with VLAN deasserted 4) Jumbo VLAN frames that were longer than the configured maximum length of a jumbo frame plus 4 in octets	3, 4
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 the configured maximum length of a jumbo frame in octets, with VLAN deasserted 4) Jumbo VLAN frames that were longer than the configured maximum length of a jumbo frame plus 4 in octets	3, 4
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).	
Total Frames Received	x‘0E’		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’	2	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.	

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. Further reception at the MAC is inhibited until the next frame starts.
4. The maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

Table 3-1. Ingress Ethernet Counters (Page 3 of 3)

Name / Group	Counter No.	Group	Description	Notes
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. Further reception at the MAC is inhibited until the next frame starts.
4. The maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

Table 3-2. Egress Ethernet Counters (Page 1 of 3)

Name	Counter No.	Group	Description	Cnt
Short Frames	x‘00’		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 octets and the configured maximum length of a jumbo frame, or between 1519 octets and configured maximum length of a jumbo frame plus 4 if VLAN is asserted. If Jumbo is not asserted, the frame is a long frame.	4

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.
4. For NP2G the maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

Table 3-2. Egress Ethernet Counters (Page 2 of 3)

Name	Counter No.	Group	Description	Cnt
Long Frames Transmitted	x'08'	4	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 the configured maximum length of a jumbo frame in octets with Jumbo deasserted 4) Jumbo VLAN frames that were longer than the configured maximum length of a jumbo frame plus 4 in octets	4
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 the configured maximum length of a jumbo frame in octets with Jumbo deasserted 4) Jumbo VLAN frames that were longer than the configured maximum length of a jumbo frame plus 4 in octets	4
Late Collisions	x'0A'		Total number of frames transmitted that experienced a network collision after 64 bytes of the frame had been transmitted.	1
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 one of the following errors: - link pointer parity errors, - header QSB field specification which point beyond the end of frame, - Invalid BCI specification . When one of these errors occurs, the frame transmission is aborted by the Egress PMM and the event is counted. Information about the event is recorded in the Aborted Frame Error Indicator register. When an Invalid BCI specification is detected, twins associated with the frame are not returned to the free queue.	

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.
4. For NP2G the maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

Table 3-2. Egress Ethernet Counters (Page 3 of 3)

Name	Counter No.	Group	Description	Cnt
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.
4. For NP2G the maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).

3.1.3 Ethernet Support

Table 3-3 lists the features and standards supported by a DMU in the different Ethernet modes.

Table 3-3. Ethernet Support

Feature	DMU Bus Mode		
	10/100 Ethernet SMII	Gigabit Ethernet GMII	Gigabit Ethernet TBI
Compatible with IEEE 802.3®, 2000.	X	X	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	X
Supports the IEEE standards on flow control by honoring received pause frames and inhibiting frame transmission while maintaining pause counter statistics.	X	X	X
Supports 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 802.3: 1993 (E).	X		
Supports full duplex point-to-point operations at media speed.	X	X	X
Detects VLAN (8100 frame type) Ethernet frames and accounts for them when calculating long frames.	X	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 device 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 device processing.	X	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	X	X
Includes jumbo frame support. When configured, can transmit and receive up to the configured maximum length of a jumbo frame in octets. This is increased by plus 4 octets when a VLAN tagged frame is detected. For NP2G, the maximum length of a jumbo frame is determined by the value in the Ethernet Jumbo Frame Size register (see *Section 13.26* beginning on page 489).	X	X	X
Supports the IBM Rx_Byte_Valid and Tx_Byte_Credit signals.		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

3.2 POS Overview

The uplink/control point port (DMU A) can support either clear-channel or channelized OC-12 when configured in packet over SONET (POS) mode.

DMU C and DMU D can support either clear-channel or channelized OC-3c, OC-12, or OC-12c POS framers.

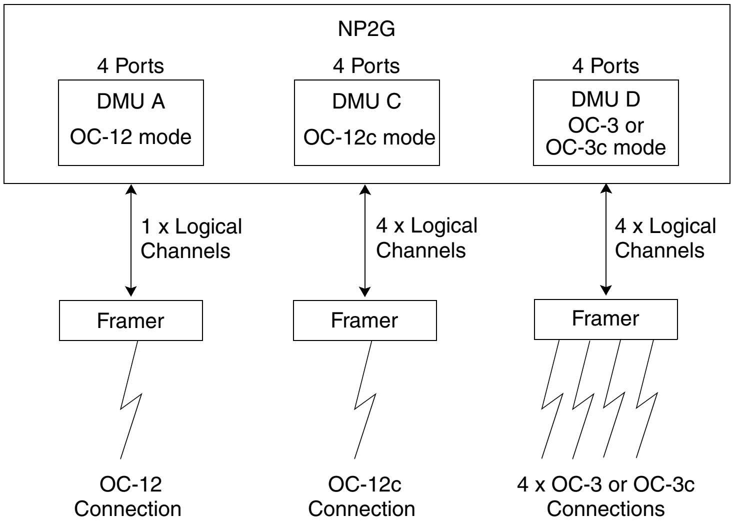
Table 3-4 shows the two DMU configuration modes: OC-3c and OC-12c. When configured for OC-3c, the DMU assumes that it must poll the four possible attached ports before transferring data. If the DMU is configured for OC-12c, the DMU does not poll the framer.

Table 3-4. DMU and Framer Configurations

DMU Configuration Mode	Networks Supported via POS Framer
OC-3c (8-bit mode)	4 x OC-3c per DMU or 1 x OC-12c per DMU
OC-12c (8-bit mode)	1 x OC-12c per DMU

Figure 3-7 shows a configuration in which the PMM has OC-3c, OC-12, and OC-3 connections operating simultaneously. Each of the three DMUs has been configured to operate as single port or as four ports. Each of the three DMUs supports an 8-bit data interface in both the ingress and egress directions.

Figure 3-7. OC-3 / OC-3c / OC-12 Configuration

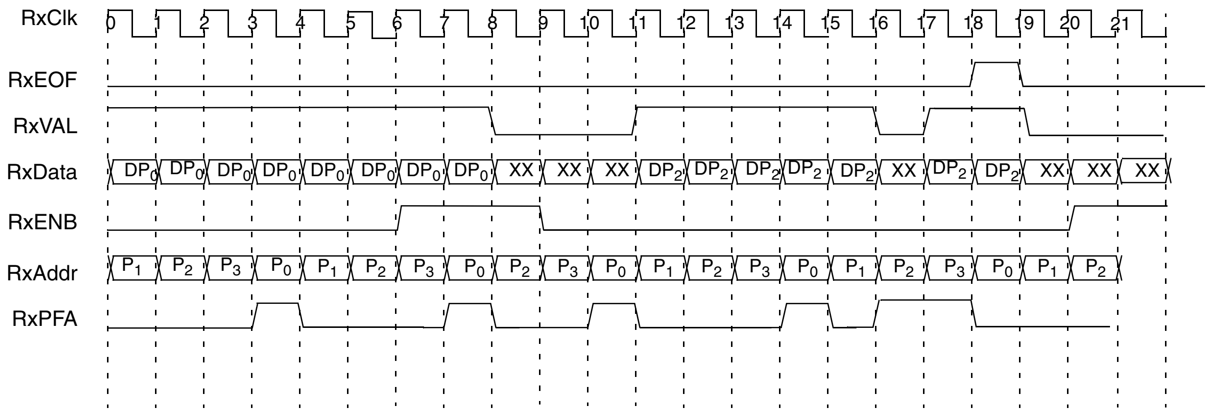


3.2.1 POS Timing Diagrams

The following figures show timing diagrams for the different POS modes:

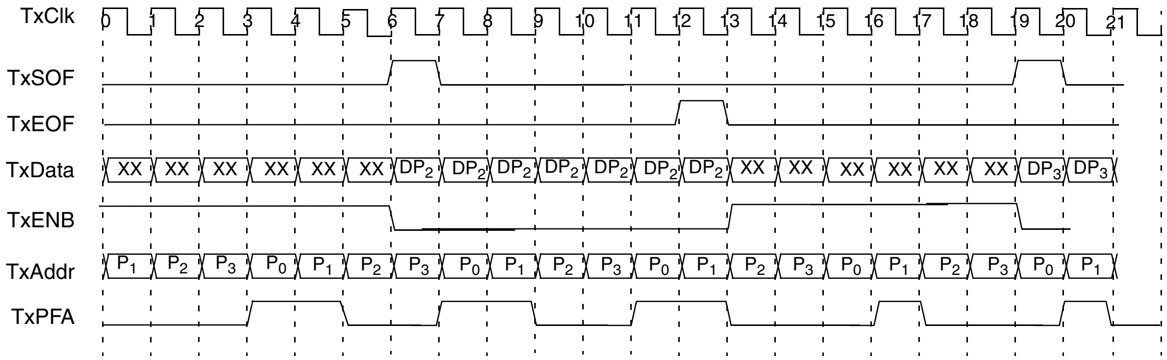
- OC-3c
  - *Figure 3-8: Receive POS8 Interface Timing for 8-bit Data Bus (OC-3c)* on page 119
  - *Figure 3-9: Transmit POS8 Interface Timing for 8-bit Data Bus (OC-3c)* on page 120
- OC-12c
  - *Figure 3-10: Receive POS8 Interface Timing for 8-bit Data Bus (OC-12c)* on page 120
  - *Figure 3-11: Transmit POS8 Interface Timing for 8-bit Data Bus (OC-12c)* on page 121

Figure 3-8. Receive POS8 Interface Timing for 8-bit Data Bus (OC-3c)


RxClk
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
RxEOF
RxVAL
RxData
DP_0 DP_0 DP_0 DP_0 DP_0 DP_0 DP_0 DP_0 XX XX XX DP_2 DP_2 DP_2 DP_2 DP_2 XX DP_2 DP_2 XX XX XX
RxENB
RxAddr
P_1 P_2 P_3 P_0 P_1 P_2 P_3 P_0 P_2 P_3 P_0 P_1 P_2 P_3 P_0 P_1 P_2 P_3 P_0 P_1 P_2
RxPFA

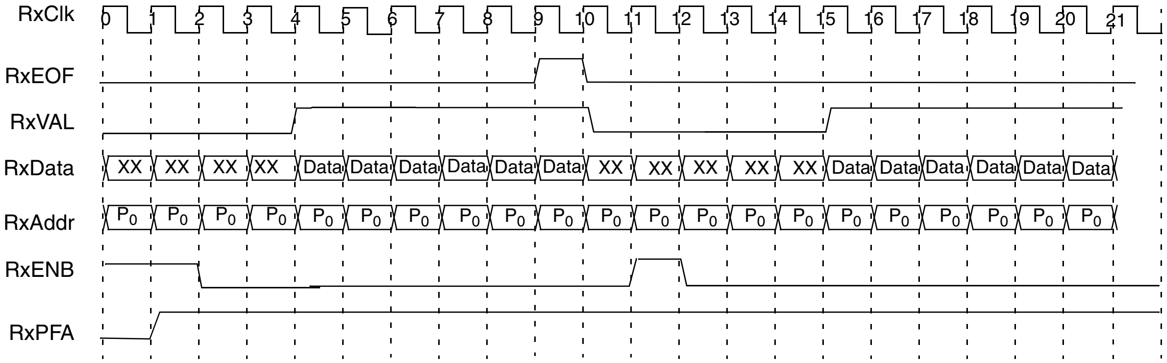
1. Data is being received from port #0 while the DMU is polling.
2. In clock cycle #3 the framer asserts RxPFA indicating data is available. In this example the DMU is configured for two cycles of framer latency (bus_delay = 1), therefore it is port #2 responding with RxPFA (clock cycle #1).
3. In clock cycle #6 the DMU deasserts RxENB (negative active) to indicate it is starting the port selection process.
4. In clock cycle #8 the DMU puts P_2 on RxAddr.
5. In clock cycle #9 the DMU selects port P_2 by asserting RxENB. The port selected is the port whose address is on RxAddr the cycle before RxENB is asserted.
6. In clock cycle #16 the framer deasserts RxVAL (assumed that the framer was not ready to continue data transfer) and restarts data transfer to the DMU during clock cycle #17.
7. In clock cycle #18 the framer asserts RxEOF marking the last transfer for the frame.
8. In clock cycle #19 the framer deasserts both RxEOF and RxVAL completing the frame transfer.
9. In clock cycle #16 and #17 RxPFA is asserted indicating data is available on ports 0 and 1.
10. In clock cycle #20 the DMU starts the selection of the next port.

Figure 3-9. Transmit POS8 Interface Timing for 8-bit Data Bus (OC-3c)


TxClk
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
TxSOF
TxEOF
TxData
XX XX XX XX XX XX DP2 DP2 DP2 DP2 DP2 DP2 DP2 XX XX XX XX XX XX DP3 DP3
TxENB
TxAddr
P1 P2 P3 P0 P1 P2 P3 P0 P1 P2 P3 P0 P1 P2 P3 P0 P1 P2 P3 P0 P1
TxPFA

1. In this figure it is assumed that the framer has a 2-cycle delay before responding to the DMU and that the DMU has bus_delay = 1.
2. In clock cycle #3 the framer responds to the Poll by asserting TxPFA, this indicates that the FIFO for ports 2 and 3 have room for data (2 clock cycle delay).
3. In clock cycle #6 the DMU starts transferring a new frame to port 2 by asserting TxSOF, asserting TxENB (negative active), and putting data on TxData.
4. In clock cycle #12 the DMU asserts TxEOF indicating the last transfer of the frame.
5. In clock cycle #13 the DMU deasserts TxENB to indicate that the DMU is in the port selection process.
6. In clock cycle #16 the framer asserts TxPFA indicating it has data for port 3.
7. In clock cycle #19 the DMU asserts TxENB and selects port 3.
8. In clock cycle #19 the DMU asserts TxSOF indicating the start of a new frame for port 3 and places data on TxData.

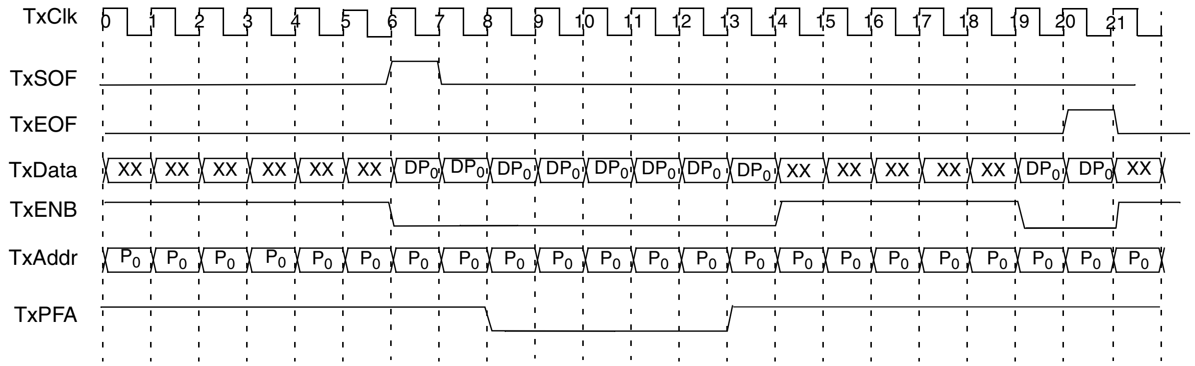
Figure 3-10. Receive POS8 Interface Timing for 8-bit Data Bus (OC-12c)


RxClk
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
RxEOF
RxVAL
RxData
XX XX XX XX Data Data Data Data Data Data XX XX XX XX XX Data Data Data Data Data Data
RxAddr
P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0 P0
RxENB
RxPFA

Note: For OC-12c, RxAddr is not provided by the DMU. RxAddr must be provided by board logic. There is a two cycle delay between the assertion of RxENB and RxVAL, the DMU can be configured to accept a one cycle delay.

1. In clock cycle #1 the framer asserts RxPFA indicating it has data to transfer to the DMU (for OC-12c operation the framer can tie RxPFA to a logical 1 and control data transfers with RxVAL).
2. In clock cycle #2 the DMU asserts RxENB (negative active) selecting the port whose address was on RxAddr during clock cycle #1 (for OC-12c operation the DMU must provide RxAddr from the card).
3. In clock cycle #4 the framer asserts RxVAL and starts the data transfer to the DMU. Data transfer continues as long as RxVAL is asserted. The framer must deassert RxVAL at the end of the frame, clock cycle #10, but can deassert and then reassert anytime within the frame.
4. In clock cycle #11 the DMU deasserts RxENB in response to RxEOF being asserted at clock cycle #9.
5. In clock cycle #12 the DMU asserts RxENB in response to RxPFA being asserted.
6. In clock cycle #15 the framer starts the transfer of the next frame by asserting RxVAL. The DMU accepts data as early as clock cycle #12 and will wait (no limit) until the framer asserts RxVAL before accepting data.
7. The framer can control data transfer using RxVAL and the DMU can control data transfer using RxENB. As noted above, RxPFA can remain asserted.

Figure 3-11. Transmit POS8 Interface Timing for 8-bit Data Bus (OC-12c)



Note: For OC-12c, TxAddr is not provided by the DMU. TxAddr must be provided by board logic.

1. In clock cycle #0 the framer has TxPFA asserted indicating it can accept data from the DMU.
2. In clock cycle #6 the DMU selects port 0 by asserting TxENB (negative active). The DMU also asserts TxSOF indicating the beginning of the frame and places data on TxData.
3. In clock cycle #8 the framer indicates it can not accept more data by deasserting TxPFA. The DMU requires 6 clock cycles to stop data transfer, hence data transfer stops after clock cycle #13 by deasserting TxEnb.
4. In clock cycle # 15 the framer asserts TxPFA indicating that it can accept more data.
5. In clock cycle #19 the DMU asserts TxENB indicating data is valid on TxData.
6. In clock cycle #20 the DMU asserts TxEOF indicating the last transfer of the frame.
7. In clock cycle #21 the DMU deasserts TxEOF and TxENB.

3.2.2 POS Counters

Table 3-5 on page 122 and *Table 3-6 on page 124* provide information about the counters maintained to support POS interfaces.

Many of the counters defined in these tables deal with long frames. A long frame is a packet whose byte count exceeds the value held in the packet over SONET Maximum Frame Size (POS_Max_FS) register (see *Configuration* on page 431). The byte count includes the cyclic redundancy check (CRC) octets.

Reception of packets meeting the criteria for long frames is 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. Further reception at the MAC is inhibited until the next packet starts.

Table 3-5. Receive Counter RAM Addresses for Ingress POS MAC (Page 1 of 2)

Port	Name	Counter Number	Description
Port 0	Long Frames Received	x‘00’	Total number of frames received 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.
Port 1 (DMU C & D only)	Long Frames Received	x‘04’	Total number of frames received 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.

Table 3-5. Receive Counter RAM Addresses for Ingress POS MAC (Page 2 of 2)

Port	Name	Counter Number	Description
Port 2 (DMU C & D only)	Long Frames Received	x‘08’	Total number of frames received 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‘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 Rx_err 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 (DMU C & D only)	Long Frames Received	x‘0C’	Total number of frames received 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‘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 3-6. Transmit Counter RAM Addresses for Egress POS MAC (Page 1 of 2)

Port	Name	Counter Number	Description
Port 0	Long Frames Transmitted	x'00'	Total number of frames transmitted 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 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 underrun occurred in the NP2G[1].
	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 one of the following errors: - link pointer parity errors, - header QSB field specification which point beyond the end of frame, - Invalid BCI specification . When one of these errors occurs, the frame transmission is aborted by the Egress PMM and the event is counted. Information about the event is recorded in the Aborted Frame Error Indicator register. When an Invalid BCI specification is detected, twins associated with the frame are not returned to the free queue.
Port 1 (DMU C & D only)	Long Frames Transmitted	x'04'	Total number of frames transmitted 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 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 underrun occurred in the NP2G[1].
	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 one of the following errors: - link pointer parity errors, - header QSB field specification which point beyond the end of frame, - Invalid BCI specification When one of these errors occurs, the frame transmission is aborted by the Egress PMM and the event is counted. Information about the event is recorded in the Aborted Frame Error Indicator register. When an Invalid BCI specification is detected, twins associated with the frame are not returned to the free queue.

[1] NP2G does not detect underrun errors; underrun detection is performed by the attached framer only.

Table 3-6. Transmit Counter RAM Addresses for Egress POS MAC (Page 2 of 2)

Port	Name	Counter Number	Description
Port 2 (DMU C & D only)	Long Frames Transmitted	x‘08’	Total number of frames transmitted 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‘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 underrun occurred in the NP2G[1].
	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 one of the following errors: - link pointer parity errors, - header QSB field specification which point beyond the end of frame, - Invalid BCI specification When one of these errors occurs, the frame transmission is aborted by the Egress PMM and the event is counted. Information about the event is recorded in the Aborted Frame Error Indicator register. When an Invalid BCI specification is detected, twins associated with the frame are not returned to the free queue.
Port 3 (DMU C & D only)	Long Frames Transmitted	x‘0C’	Total number of frames transmitted 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‘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 underrun occurred in the NP2G[1].
	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 one of the following errors: - link pointer parity errors, - header QSB field specification which point beyond the end of frame, - Invalid BCI specification . When one of these errors occurs, the frame transmission is aborted by the Egress PMM and the event is counted. Information about the event is recorded in the Aborted Frame Error Indicator register. When an Invalid BCI specification is detected, twins associated with the frame are not returned to the free queue.

1. NP2G does not detect underrun errors; underrun detection is performed by the attached framer only.

3.2.3 POS Support

Table 3-7 lists the features and standards supported by a DMU in the different POS modes.

Table 3-7. POS Support

Feature	POS Mode
	OC-3c & OC-12c (8-Bit)
Supports quad-port OC-3c connections, quad-port OC-12 connections, or single-port OC-12c connections	X
Compatible with Flexbus™ 3 - Quad- 8 bit bus operational mode	X
Compatible with Flexbus™ 3 - Single - 32 bit bus operational mode	
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
The minimum frame length the NP2G 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
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

4. Ingress Enqueuer / Dequeuer / Scheduler

4.1 Overview

The Ingress Enqueuer / Dequeuer / Scheduler (Ingress EDS) interfaces with the Physical MAC Multiplexer (PMM), the Embedded Processor Complex (EPC), and the Ingress-to-Egress Wrap (IEW). Frames that have been received on the Data Mover Unit (DMU) interface are passed through the PMM to the Ingress EDS. The Ingress EDS collects the frame data in its internal Data Store; when it has received sufficient data, the Ingress EDS enqueues the frame to the EPC for processing. The Ingress EDS does not need to receive the entire frame before it enqueues the data (that is, it can operate in cut-through mode).

Once the EPC processes the frame, it provides forwarding and quality of service (QoS) information to the Ingress EDS. The Ingress EDS then invokes hardware configured flow control mechanisms and then either discards the frame or places it into a queue to await transmission.

The Ingress EDS schedules all frames that cross the IEW. After it selects a frame, the Ingress EDS passes the frame data to the IEW, which segments the frame into data cells and sends them on to the Egress EDS.

4.2 Operation

Figure 4-1. Logical Organization of Ingress EDS Data Flow Management

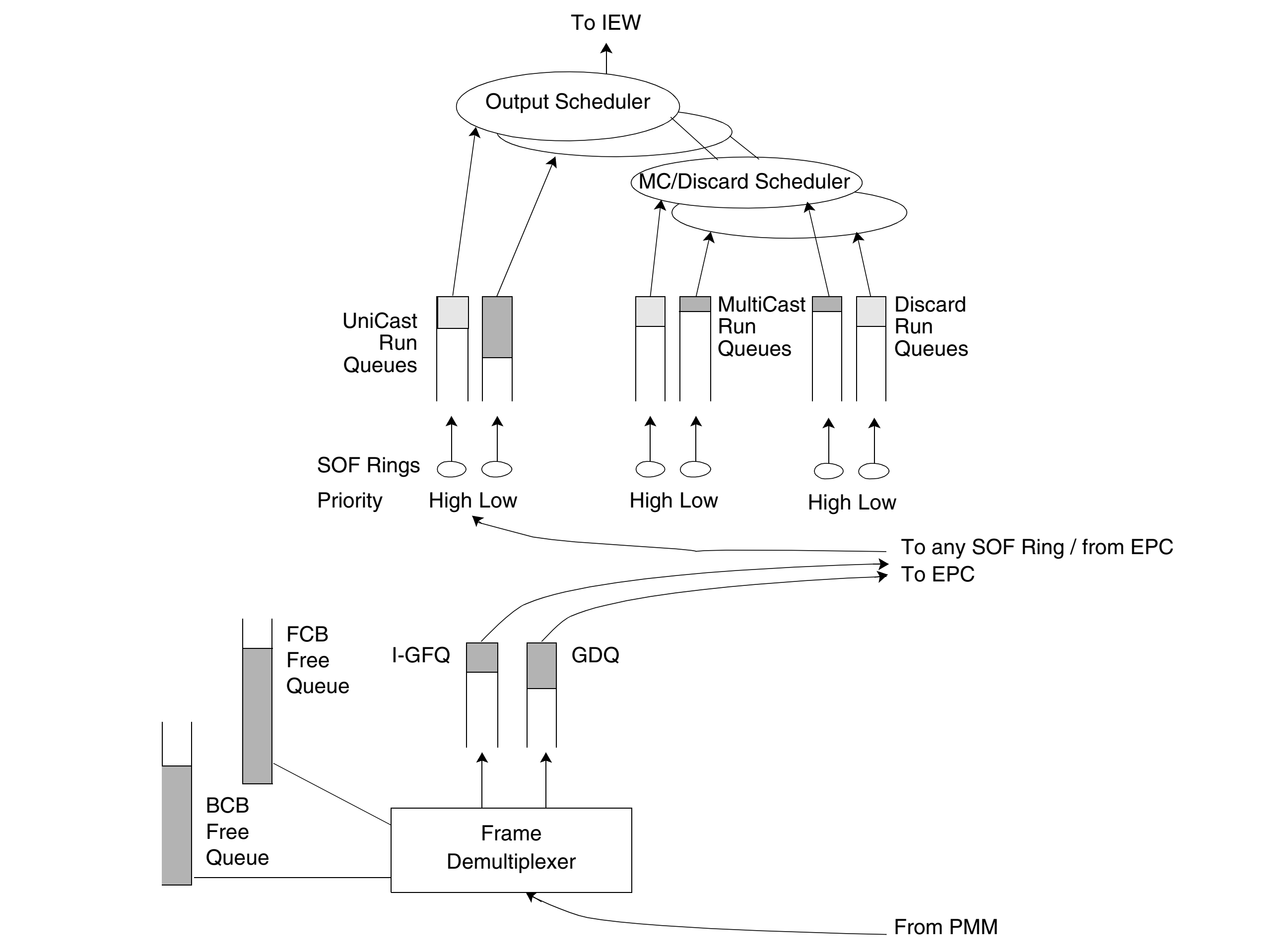


Figure 4-1 illustrates the operations of the Ingress EDS discussed in this section.

The Frame Demultiplexer (FD) receives incoming frames from the PMM. An FCB is allocated on-the-fly from the FCB Free queue as each new frame starts. The FD writes each frame as a list of chained data buffers into its internal ingress Data Store. The buffer addresses are dequeued on-the-fly from the BCB Free queue, which is the linked list of all currently available data buffer addresses.

The EPC can process the frames under one of the following conditions:

- Cut-through mode is disabled and the frame has been completely received.
- Cut-through mode is enabled and sufficient data has been received.
  The quantity of sufficient data in cut-through mode is programmable in increments of 64 bytes. The first 64 bytes of a frame, in most cases, contain the information to be used in Layer 2, Layer 3, and Layer 4 forwarding.

When a frame is ready for processing by the EPC, the Ingress EDS enqueues its FCB to an input queue: either the I-GFQ (for control frames) or the GDQ (for data frames).

The EPC is responsible for recognizing that a frame is ready to be processed, dequeueing and processing it, and giving it back to the Ingress EDS (by enqueueing to a TDMU queue) for forwarding to the IEW.

The EPC returns each frame in an ingress enqueue operation to a specific target Data Mover Unit (TDMU) queue (as determined by the routing table lookup). At this point in the flow, the Ingress EDS flow control logic determines if the frame should be enqueued into the Ingress Scheduler (that is, added to the TDMU queue) or if it should be discarded. See *Section 4.3 Ingress Flow Control* on page 132 for details.

TDMU queues are linked in start-of-frame (SOF) rings. An SOF ring can hold up to four TDMU queues. To maintain optimal ring structure, only non-empty TDMU queue are included in an SOF ring.

Each SOF ring is associated with a particular Run queue, creating an SOF ring / Run queue set that provides a fair initiation of frame transmission to all DMUs of the Network Processor. Fairness is achieved by a round-robin mechanism that successively selects each TDMU queue in the SOF Ring. When a TDMU queue is selected, the frame currently at its head is dequeued from the TDMU queue and enqueued in a Run queue (the actual entry-point of data movement to the IEW once sufficient data for forwarding has been received.

There are two SOF ring / Run queue sets: one for Unicast High Priority Traffic and one for Unicast Low Priority Traffic. Four other sets are reserved for Multicast High and Low Priority Traffic, and for High and Low Priority Discard Traffic.

For each priority plane, round-robin scheduling is performed between the selected unicast candidate and the candidate from the multicast and discard Run queues.

These schedulers are controlled by the status (empty or non-empty) of each Run queue, and by the flow control indication received from the Egress side.

Absolute priority is enforced: the Low Priority candidate is used only if there is no High Priority candidate present at the time.

The cut-through function is supported by queueing partial frames that contain enough data to fill a cell. If, after scheduling to the IEW, enough data remains in the frame to fill another cell, then the frame is kept in the Run queue. If all the frame data has not been received from the PMM, but there is insufficient data in the data store to fill the next cell, the FCB is removed from the Run queue and left “floating” (not part of any queue). When the Frame Demultiplexer detects that there is enough data to fill a cell, or the last byte of the frame is received, it re-enqueues the frame in its Run queue. If all the frame data has been received, the FCB stays in the Run queue until all the data has been moved to the IEW. At that point, the FCB is sent to the FCB free queue.

As can be seen from the above description, the I-EDS may have multiple frames “in flight”. The number of frames that may be simultaneously in a state of transmission is controlled by the number of correlators available, which is 64 for unicast and 32 for multicast. The I-EDS maintains correlator free pools for unicast and multicast frames and for each priority (total of four free pools).

Note: For packed frames, two correlators are required, one for the frame completing and one for the frame that is starting (see *Table 5-1: Cell Header Fields* on page 137 and *Table 5-2: Frame Header Fields* on page 139).

4.2.1 Operational Details

This section provides more details about some of the operations described in *Section 4.2*.

Figure 4-2 details the structure of the start-of-frame (SOF) rings acting as multiple-port entry points to the target Data Mover Unit (TDMU) queues. An SOF ring is a circular linked list of TDMU queues dynamically inserted into and removed from the ring. Note that the multicast and discard SOF rings contain only a single TDMU queue.

Figure 4-2. SOF Ring Structure

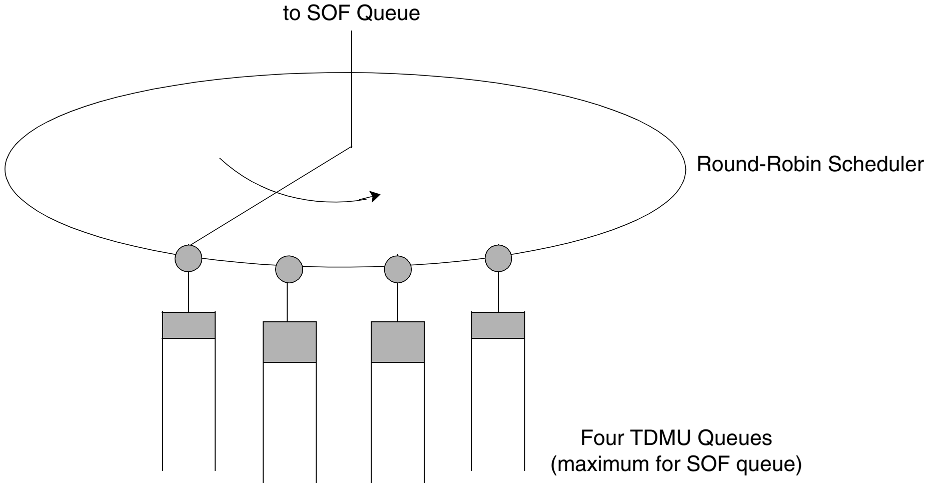


Figure 4-3: Ingress EDS Logical Structure on page 131 illustrates the various logical structures of the EDS that are discussed in the rest of this section.

A frame (or packet) is stored in the internal Ingress Data Store in the form of a linked list of data buffers. Each data buffer is a 64-byte area of the Ingress Data Store. The Ingress Data Store has 2048 data buffers available, each with a unique identifier value (0 through 2047).

Data buffers are chained by Next Buffer Control Block Address (NBA) pointers located in buffer control blocks (BCBs). One BCB is associated with each data buffer and shares the same identifier value (there are 2048 BCBs). BCBs are physically located in an independent imbedded memory.

Each frame is represented by a frame control block (FCB) that identifies the new frame by data location (the address of the first buffer in the Data Store) and control information (the Source Port from which this frame originated). 2048 FCBs are available in an independent embedded memory. Each FCB contains parameters associated with the frame, such as the two pointers shown in *Figure 4-3*:

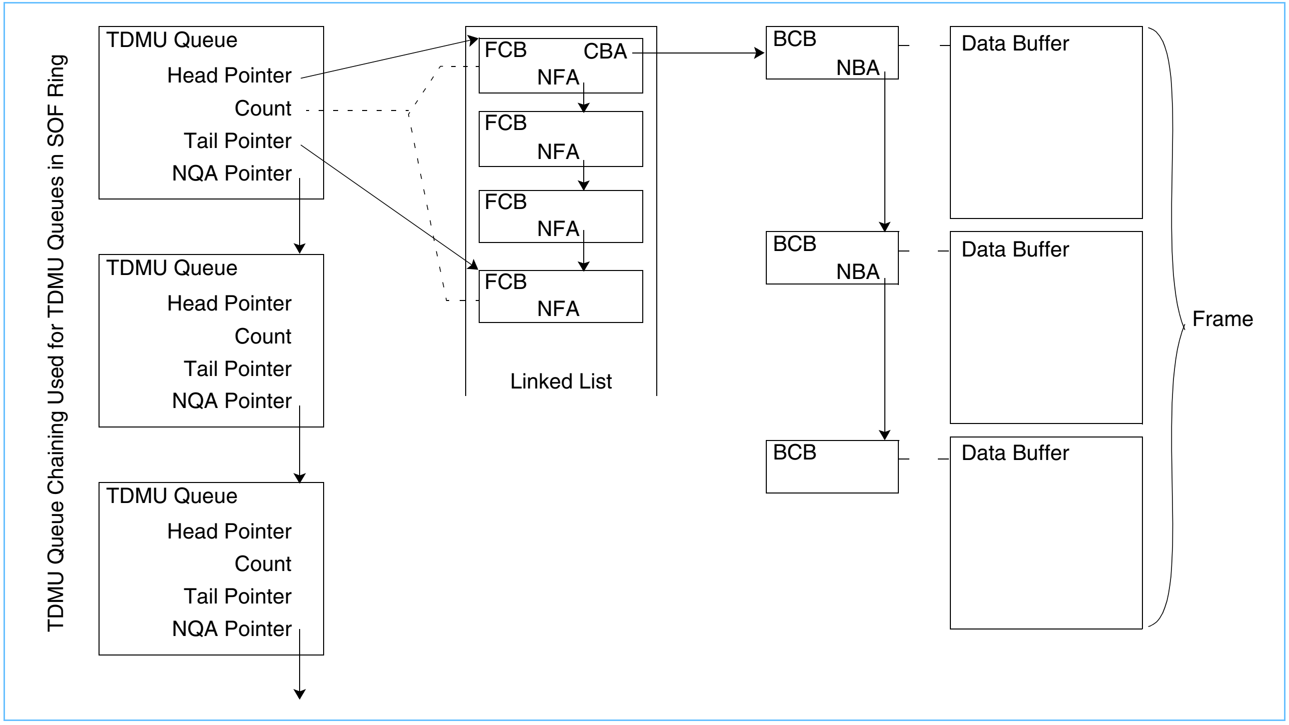
- The control block address (CBA) points to the first data buffer of a frame (for frames whose transmission to the IEW has not started), or the data buffer ready to be transmitted (for frames whose transmission to the IEW has already started).
- The next FCB address (NFA) points to the FCB of the next frame in the queue.

The FCB also maintains information about the amount of one frame's data currently in the Ingress Data Store. As the data is moved out of the Data Store across the IEW, the FCB accounting information reflects the reduction in data present in the Data Store. At the same time, more of the frame's data may still be arriving from the PMM, and the accounting information in the FCB will reflect the increase of data in the data store.

Frames are chained in queues which consist of FIFO-linked lists of FCBs and the queue control blocks TDMU queues. A TDMU queue contains the information necessary to manage the linked list of FCBs, particularly:

- Head pointer: points to the first FCB in the queue
- Tail pointer: points to the last FCB in the queue
- Count: indicates how many FCBs are present in the queue
- Next QCB Address (NQA) pointer: points to the next TDMU queue. This pointer enables the chaining of TDMU queues and is the mechanism for creating the SOF ring and TDMU queues.

Figure 4-3. Ingress EDS Logical Structure



4.3 Ingress Flow Control

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 the egress side, and so on. A transmit probability for various flows is an output of these algorithms.

The network processor implements flow control in two ways:

- Flow control is invoked when the frame is enqueued to a start-of-frame (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 is invoked when frame data enters the network processor. When the Ingress DS 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 4-1: Flow Control Hardware Facilities* on page 133 for more information).

4.3.1 Flow Control Hardware Facilities

The hardware facilities listed in *Table 4-1* on page 133 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 4-1. 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, 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. A second threshold (BCB_FQ_Th_0_SA) can be configured which the BCB_FQ must be exceeded before buffers will start being allocated again for incoming user traffic after the BCB_FQ has dropped below the BCB_FQ_Threshold_0 value (see *BCB_FQ_Threshold_0 Register (BCB_FQ_TH_0)* on page 456).	CAB
BCB_FQ_Threshold_1	Threshold for BCB_FQ. When BCB_FQ < BCB_FQ_Threshold_1, 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. A second threshold (BCB_FQ_Th_1_SA) can be configured which the BCB_FQ must be exceeded before buffers will start being allocated again for new incoming user traffic after the BCB_FQ has dropped below the BCB_FQ_Threshold_0 value (see *BCB_FQ_Threshold_0 Register (BCB_FQ_TH_0)* on page 456).	CAB
BCB_FQ_Threshold_2	Threshold for BCB_FQ. When 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, the I_FreeQ_Th is set to 1. This may be used by external devices assisting in flow control.	CAB
Unicast/Multicast Status 0	Threshold status of the priority 0 TDMU queues and the multicast queue. The count and the threshold are compared for each DMU. The results are combined into a single bit result. If TDMU_QCB.QCnt > TDMU_QCB.Th, the bit is set to 1; if not, it is set to 0.	CAB
Egress Status 0	This 1-bit register contains the congestion status of the Egress Data Stores. The congestion status is the result a comparison between the configured threshold and the EWMA of the offered rate of priority 0 traffic (see *Table 6-1: Flow Control Hardware Facilities* on page 148).	Hardware only
Egress Status 1	This 1-bit register contains the congestion status of the Egress Data Stores. The congestion status is the result a comparison between the configured threshold and the EWMA of the offered rate of priority 1 traffic (see *Table 6-1: Flow Control Hardware Facilities* on page 148).	Hardware only

4.3.2 Hardware Function

4.3.2.1 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 $\in$ {1, 1/2, 1/4, 1/8}.

4.3.2.2 Flow Control Hardware Actions

When the picocode enqueues a packet to be transmitted to the egress side, the flow control hardware examines the state of the FQ_P0_Th and FQ_P1_Th threshold statuses 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, Egress Status 0 is exceeded, the packet is discarded (tail drop discard). The picocode must set up a counter block to count these discards.
- For priority 1 packets, if FQ_P1_TH is exceeded, the packet is discarded (tail drop discard). Otherwise, the transmit probability table is accessed and the value obtained is compared against a random number ($\in$ {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 Quality of Service (QoS) Class taken from the DSCP (Ingress FCBpage TOS field bits 7:5).

  T Egress Status; one bit corresponding to the egress congestion (zero when the frame is multicast).

  CC Diffserv code point (DSCP) assigned color (Ingress FCBpage FCInfo field bits 1:0).

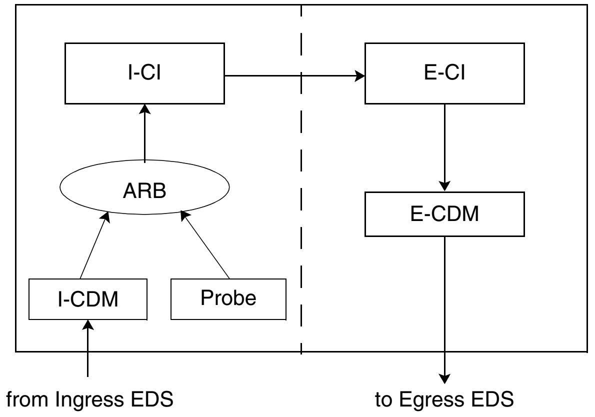
5. Ingress-to-Egress Wrap

The Ingress-to-Egress Wrap (IEW) provides an interface to connect the ingress side of the NP2G to the egress side. It supports 3.25 to 4 Gigabits per second (Gbps) throughput. An arbiter sits between the Ingress Cell Data Mover (I-CDM) and Ingress Cell Interface (I-CI). The arbiter directs the data traffic from the I-CDM and Probe to the I-CI. *Figure 5-1* shows the main functional blocks of the IEW.

The IEW supports the following:

- Building a Cell Header and Frame Header for each frame
- Segmenting a frame into 64-byte cells
- Cell packing

Figure 5-1. Ingress-to Egress Wrap Functional Blocks



The main units, described in following sections, are:

- Ingress Cell Data Mover (I-CDM)
- Ingress Cell Interface (I-CI)
- Egress Cell Interface (E-CI)
- Egress Cell Data Mover (E-CDM)

5.1 Ingress Cell Data Mover

The Ingress Cell Data Mover (I-CDM) is the logical interface between the Ingress Enqueuer / Dequeuer / Scheduler's (EDS) frame data flow and the IEW's cell data flow. The I-CDM segments the frames into cells and passes the cells to the I-CI. The I-CDM also provides the following frame alteration functions:

- VLAN insert or overlay. The four bytes comprising the VLAN type field (x‘8100’) and the Tag control field are placed after the 12th byte of the frame.

- n-byte delete. Bytes may be deleted from the incoming frame prior to being sent to the egress side.

The Ingress EDS controls the I-CDM data input by requesting the I-CDM to transmit data to the egress side. To do this, the Ingress EDS gives the I-CDM 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 of the same priority (used for packing). With this information, the I-CDM builds a cell under a quadword-aligned format compatible with the data structure in the Egress Data Store. The logical 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 through the IEW, 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 with unicast frames when the priority of the next frame is the same as the 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.

To complete the logical cell, the I-CDM assembles control information from the FCB that was sent from the Ingress EDS, and utilizes data bytes read from the ingress data store buffers. Logical cells are passed to the I-CI using a grant handshake.

5.1.1 Cell Header

A cell header is a 6-byte field holding control information for the cell.

A cell header is sent with each cell through the IEW. *Figure 5-2* illustrates the format of a cell header; *Table 5-1* on page 137 provides the field definitions.

Figure 5-2. Cell Header Format

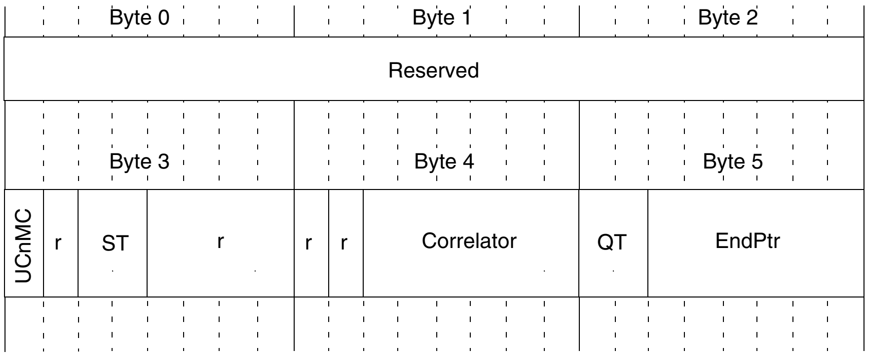


Table 5-1. Cell Header Fields

Field Name	Definition	Notes
UCnMC	Unicast - Multicast indicator. This 1-bit field indicates the format of the frame carried in this cell. ‘0’ Multicast format. Guided traffic must be sent in multicast format. ‘1’ Unicast format.	1
ST(1:0)	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 (the cell contains a middle portion of a frame, but no start or end of a frame) 01 End of current frame. This code point must be used when the Cell format value is '11' (packed frame format). (The cell contains the end of frame if “frame format” is indicated by the cell format value, or the end of one frame and the beginning of another frame if “packed frame format” is indicated.) 10 Start of new frame (the cell contains the start of a frame, but no end of a frame) 11 Start and end of new frame. Used for frames 48 bytes or smaller. This code point is also used to indicate a reassembly abort command. Abort is indicated when the End Pointer field value is ‘0’. An aborted frame is discarded by the E-EDS. At the point of reassembly, cells for a frame are expected to follow a start, continuation (may not be present for short frames) and end sequence. Hardware in the E-EDS detects when this sequence is not followed and reports these errors in the Reassembly Sequence Error Count Register. Frames with reassembly sequence errors are discarded.	
Correlator	Correlator values 0 - 63 are used for unicast traffic. Correlator values 0- 31 are used for multicast traffic.	1
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 a data queue (GRx or GBx). 1x Guided traffic that is enqueued into the guided frame queue. x0 Cell may be dropped due to congestion. x1 Cell may not be dropped due to congestion. QT(0) is set to 1 by the IEW hardware when FC Info field of the frame header is set to x‘F’.	
EndPtr	End Pointer (6-bit field). The EndPtr field is a byte offset within the cell which indicates the location of the last data byte of the current frame in the cell when the ST field indicates end of current frame (ST = ‘01’ or ‘11’). Valid values of the EndPtr field are in the range of 6 through 63. An EndPtr value of 0 is used only with an ST value of '11' to indicate an abort command. Values of 1 through 5 are reserved. In all other cases (ST = ‘00’ or ‘10’), this field contains sequence checking information used by the frame reassembly logic in the network processor. The sequence checking information consists of a sequence number placed into this field. The first cell of a frame is assigned a sequence number of 0. In a packed cell, the sequence number is not placed into the EndPtr field since the field must contain the end pointer for the preceding frame; the next cell for this frame will contain a sequence number of 1. Sequence numbers increment from 0 to 63 and will wrap if necessary.	
r	Reserved field, transmitted as ‘0’.	

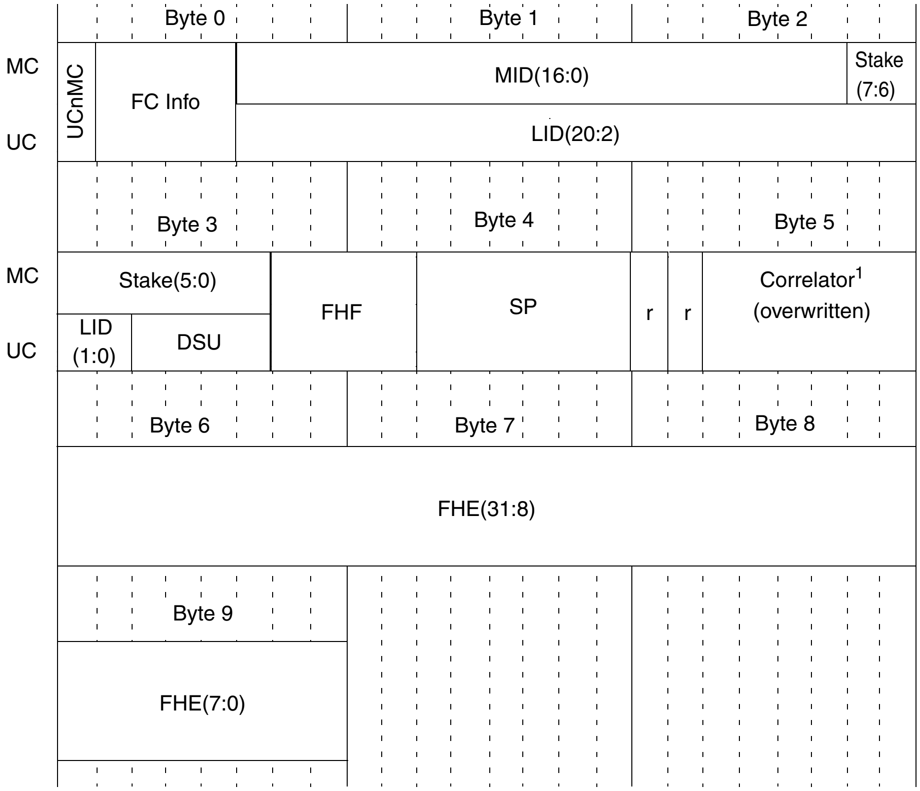
1. This field is used by the egress reassembly logic in selection of the reassembly control block.

5.1.2 Frame Header

A frame header is a 10-byte field containing control information used in egress processing and is sent once per frame. A frame header can immediately follow the cell header, or with packing enabled, it can also be placed at the start of the 2nd (D10), 3rd (D26), or 4th (D42) QW of the cell.

Figure 5-3 illustrates the format of a frame header; *Table 5-2* on page 139 provides the field definitions.

Figure 5-3. Frame Header Format



1. See description of the correlator field in *Frame Header Fields* on page 139.

Table 5-2. Frame Header Fields

Field Name	Description	Notes
UCnMC	Unicast - Multicast indicator. This 1-bit field indicates the format of the frame header. ‘0’ Multicast format. ‘1’ Unicast format.	1
FC Info	Flow Control Information (4-bit field). Indicates the type of connection used for this frame. Connection type encoding is used by the NP2G’s flow control mechanisms.	2
LID	Lookup Identifier (21 bit field). Used by the egress processing to locate the necessary information to forward the frame to the appropriate target port with the correct QoS.	3
MID	Multicast Identifier (17 bit field). 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.	3
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.	3
DSU	DSU indicator. Available only for the unicast format of the frame header. This 4-bit field indicates which egress data stores are used by this frame when toggle mode is disabled (see *13.2 Toggle Mode Register* on page 444). 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.	
r	Reserved field, transmitted as ‘0’.	
Correlator	Correlator values 0-63 are used for unicast traffic. Correlator values 0-31 are used for multicast traffic.	

1. This field is used by egress reassembly logic in selection the reassembly control block. It is also used by the hardware classifier.
2. This field is passed through the ingress flow control before the frame is enqueued for transmission through the IEW. It can be used by software to allow the hardware to make further flow control decisions on egress by passing connection information (type and DSCP).
3. This field is passed by the hardware but is defined and used by the software.

5.2 Ingress Cell Interface

The Ingress Cell Interface (I-CI) performs formatting on the cell headers to conform with the cell format previously defined.

The I-CI unit relies on an internal FIFO buffer (written by the I-CDM at the network processor core clock rate and read at the IEW clock rate).

5.3 Egress Cell Interface

The Egress Cell Interface (E-CI) receives cells from the IEW and passes data cells to the Egress Cell Data Mover (E-CDM). The E-CI assists in egress data flow control by throttling the ingress traffic to prevent data overruns. It does this by passing the egress congestion status back to the ingress scheduler.

5.4 Egress Cell Data Mover

The Egress Cell Data Mover (E-CDM) is the logical interface between the cell data flow of the E-CI and the frame data flow of the Egress EDS. The E-CDM serves as a buffer for cells flowing from the E-CI to the Egress EDS. The E-CDM 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-CDM, to re-assemble the cells into frames.

6. Egress Enqueuer / Dequeuer / Scheduler

The Egress Enqueuer / Dequeuer / Scheduler (Egress EDS) interfaces with the Physical MAC Multiplexer (PMM), the Embedded Processor Complex (EPC), and the Ingress-to-Egress Wrap (IEW). It 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 reassembles data that have been transported from the ingress side. Each cell received from the IEW 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 IEW, the Egress EDS enqueues it to the EPC for processing.

Once the EPC processes the frame, it provides forwarding and quality of service (QoS) information to the Egress EDS. The Egress EDS then invokes hardware configured flow control mechanisms and then enqueues it to either the Scheduler, when enabled, or to a target port (TP) queue for transmission to the Egress PMM (which, in turn, sends the data to physical layer devices).

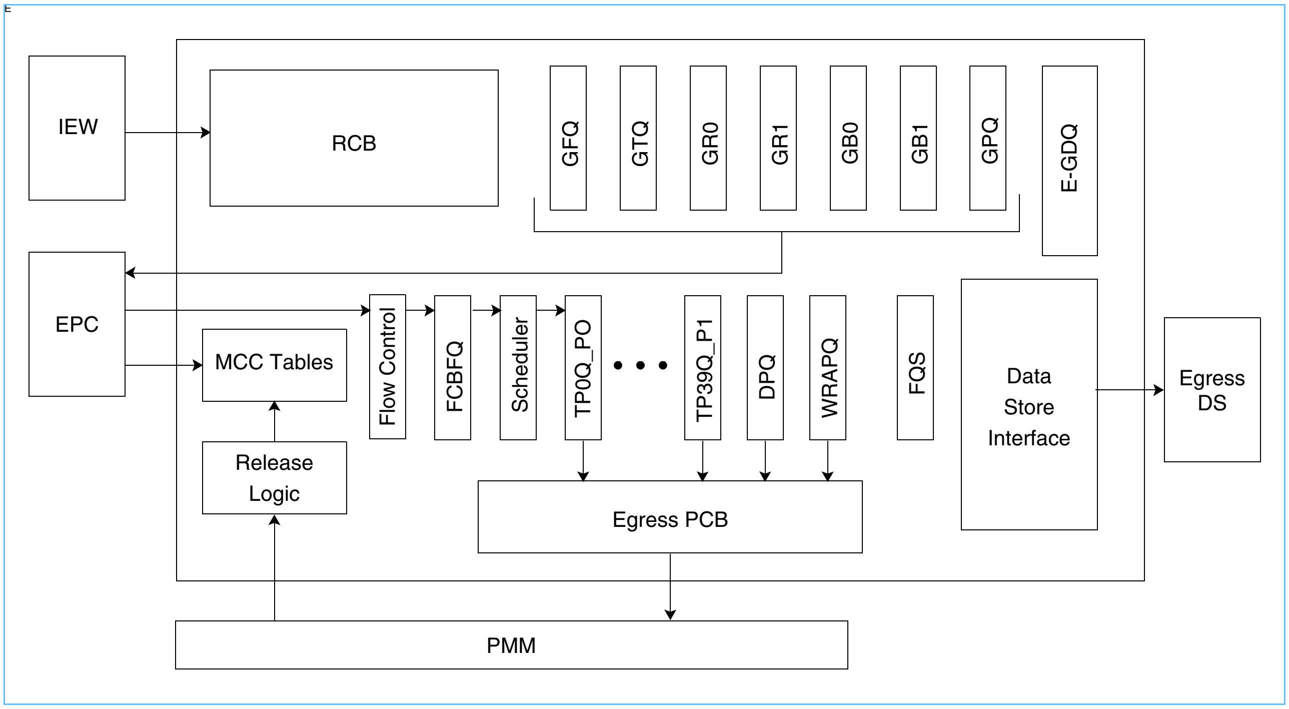
The Egress EDS supports the following functions, as illustrated in *Figure 6-1* on page 142:

- 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 frame control blocks (FCBs) depending on memory configuration
- Unicast and multicast frames
- Cell packing
- Up to 21 external ports plus wrap port interface to the PMM
- Discard function
- Hardware initialization of internal and external data structures

6.1 Functional Blocks

Figure 6-1 illustrates the functional blocks of the Egress EDS. The list following the figure explains each functional block.

Figure 6-1. Egress EDS Functional Blocks



Data Store Interface — Writes the external Egress DS during frame reassembly and reads them during frame transmission. Also gives the EPC access to the Egress DS. 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. The picocode uses this queue to discard frames where header twins have been allocated.

E-GDQ — Discard Queue Stack. Holds frames that need to be discarded. The hardware uses this queue to discard frames when the egress DS is congested or to re-walk a frame marked for discard for a half-duplex port. The picocode uses this queue to discard frames that do not have header twins allocated.

Egress PCB — Egress Port Control Block. Contains the necessary information 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 so on. 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 155) or a TP 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 obtains a new free twin buffer any time the Egress EDS needs one and returns 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 General PowerPC Handler (GPH) thread 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 picocode for delivery to the General Table Handler (GTH) thread.

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 ingress side 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 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 TP. Two queues are associated with each of the 21 possible TPs. 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 are typically used for debug or diagnostic purposes.

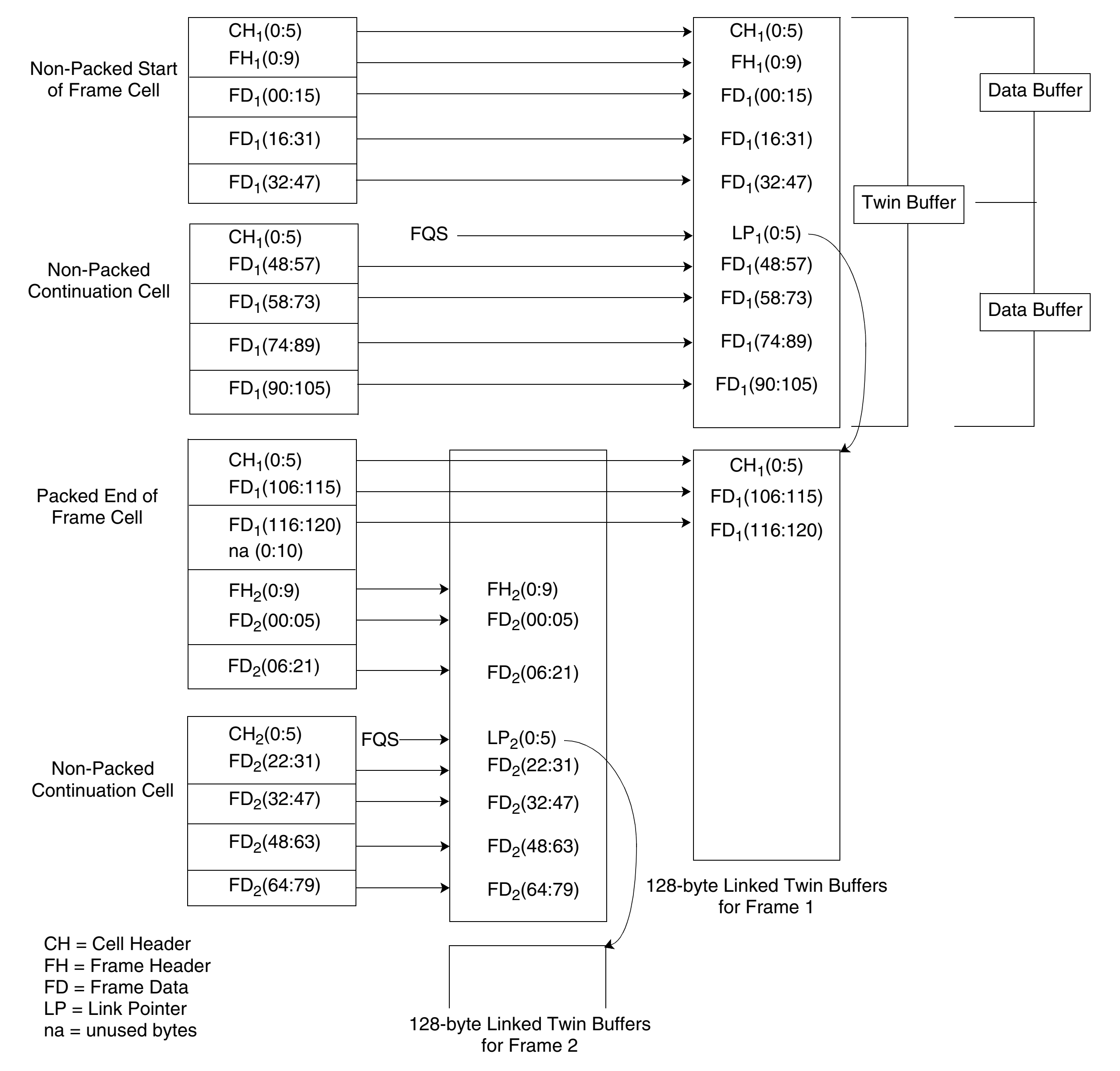
6.2 Operation

The Egress EDS receives cells from the Ingress-to-Egress Wrap (IEW) along with information that has been preprocessed by the Egress Cell Data Mover (CDM) such as the Reassembly Correlator, multicast indication, priority, and target data store (DS). 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 cell (64 bytes) in external DRAM). Cell window time is configured using the DRAM Parameter Register’s 11/10 field.

The Egress EDS reads the cell data out of the CDM and uses the Reassembly Correlator, multicast indication, and priority information to index into the Reassembly Control Block (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 the same frame. The Egress EDS manages 192 RCB entries, and each entry contains information such as start-of-frame indicator, data store buffer address, current reassembled frame length, and queue type. The Egress EDS uses the RCB information to rebuild each frame.

The Egress EDS uses a free buffer from the head of the free queue stack (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 is the beginning of another frame and this data is stored in a second twin buffer as indicated by the second frame’s RCB entry. *Figure 6-2* illustrates the cell buffers and storage in an Egress DS.

Figure 6-2. Cell Formats and Storage in the Egress DS

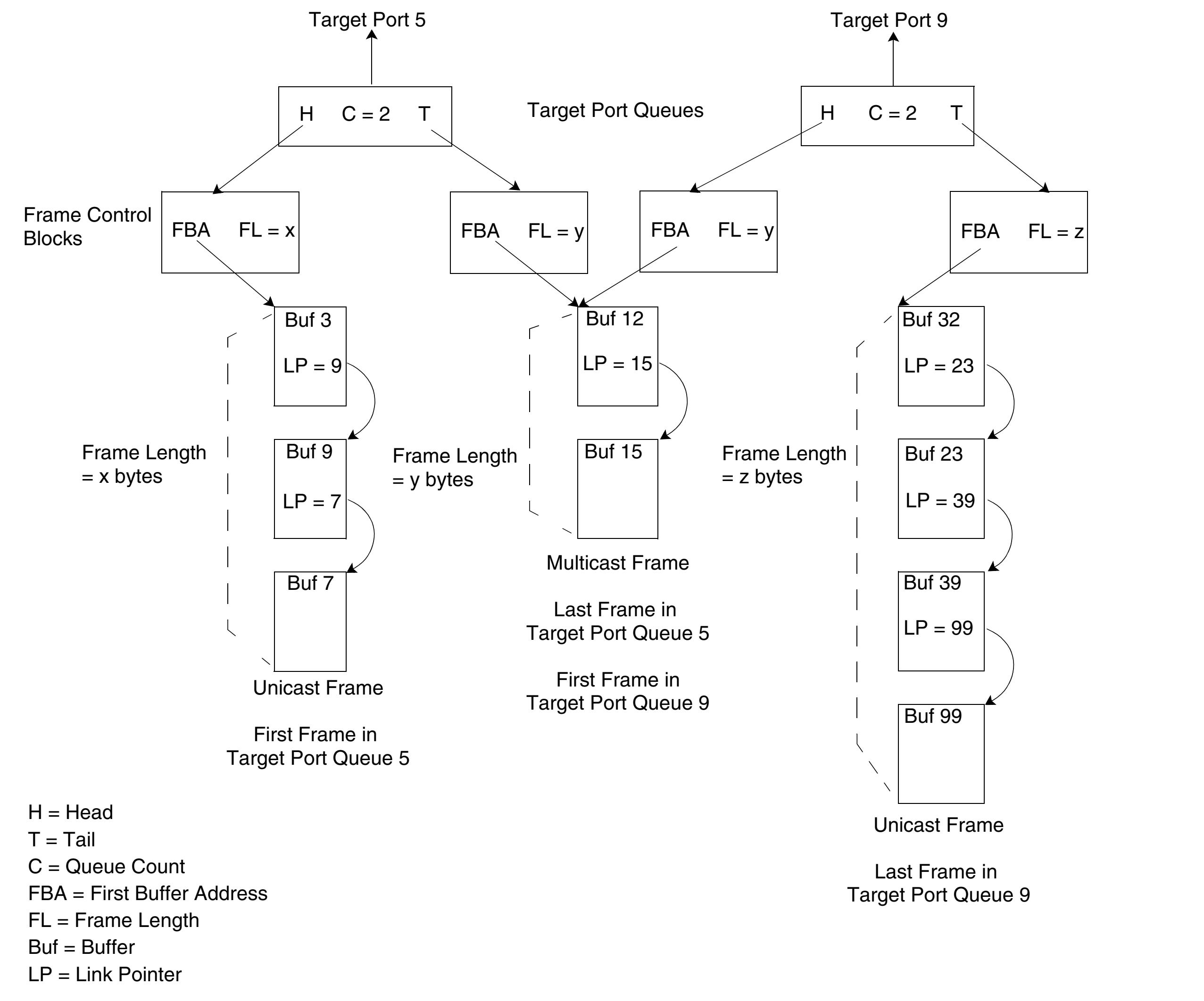


When the entire frame is reassembled, the Egress EDS enqueues it to one of several EPC queues. If the reassembled frame is a guided frame, the Egress EDS uses the GFQ. High priority frames are placed in either the GR0 or the GR1. Low priority frames are placed in either the GB0 or the GB1. The EPC services these queues and requests a programmable amount of read data from the various frames in order to process them (see *Table 7-84: Port Configuration Memory Content* on page 262). 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 (GTH), 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 (if the scheduler is disabled) or into a QCB configured for wrap traffic. 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, as illustrated in *Figure 6-3*. 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 6-3. 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 (PCB) entry for that port and the FCB for that frame is placed on the FCB Free Queue. The Egress EDS uses the PCB to manage frames being sent to the PMM for transmission. The PCB stores information needed to retrieve frame data, such as current buffer address and frame length, from the Egress DS. The PCB allows up to 21 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 transfers and stores 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 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

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, and so on. A transmit probability for various flows is an output of these algorithms.

The network processor implements flow control in two ways:

- Flow control is invoked when the frame is enqueued to either a target port queue or a flow queue control block (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 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 6-1: Flow Control Hardware Facilities* on page 148 for more information).

6.3.1 Flow Control Hardware Facilities

The hardware facilities listed in *Table 6-1* 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 6-1. Flow Control Hardware Facilities (Page 1 of 2)

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	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	The calculated EWMA of the Free Queue Count.	CAB
P0 Twin Count	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
P1 Twin Count	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. It 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. It 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	The calculated EWMA based on 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	The calculated EWMA based on 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. It 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 the ingress side via the res_data I/O and is collected in the Status 0 register in the ingress flow control hardware.	CAB

Table 6-1. Flow Control Hardware Facilities (Page 2 of 2)

Name	Definition	Access
Egress P1 Twin Count EWMA Threshold	The congestion status of the Egress Data Store. It 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 the egress side via the res_data I/O and is collected in the Status 1 register in the ingress flow control hardware.	CAB
Target Port TH_PQ+FQ	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 all ports. 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

6.3.2 Remote Egress Status Bus

The Remote Egress Status (RES) Bus communicates the congestion state of the Egress Data Stores to the Ingress Flow Control. The ingress portion of each NP2G can then preemptively discard frames to reduce consumption of egress bandwidth.

6.3.2.1 Bus Sequence and Timing

The RES bus passes status information from the egress side of the NP2G back to the ingress side. It consists of a data signal (RES_Data) and a synchronous signal (RES_Sync), both of which can be monitored by external logic.

Figure 6-4 on page 149 shows a timing diagram of the operation of the RES Bus.

Figure 6-4. RES Bus Timing

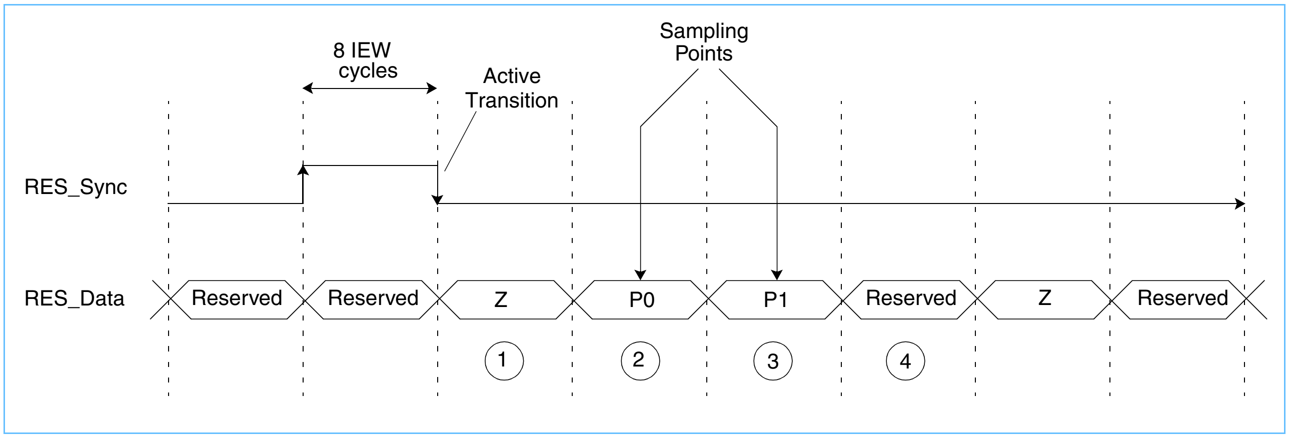


Figure 6-4 illustrates the timings used for Res_Sync and Res_Data. Res_Sync is shown positive active (the falling transition is used to determine where the Res_data bus sequence starts).

The RES Bus operates on the same clock frequency as the internal IEW clock. This clock period can range from 8 ns to 10 ns. However, the RES Bus clock is not necessarily in phase with the IEW clock.

The NP2G puts four values on the RES_Data signal. Each value is held for eight IEW clock cycles. Therefore, each NP2G time slot is 32 IEW clock cycles. The protocol for sending the congestion information is as follows:

1. **High-Z**. The NP2G keeps its RES_Data line in high impedance for eight IEW clock cycles. This allows the bus to turn around from one NP2G to another.
2. **P0**. The NP2G drives its Priority 0 (P0) Egress Data Store congestion information for eight IEW clock cycles. Congestion is indicated when the Egress P0 Twin Count EWMA register value is greater than the Egress P0 Twin Count EWMA Threshold register value.
3. **P1**. The NP2G drives its Priority 1 (P1) Egress Data Store congestion information for eight IEW clock cycles. Congestion is indicated when the Egress P1 Twin Count EWMA register value is greater than the Egress P1 Twin Count EWMA Threshold register value.
4. **Reserved**. The NP2G drives a low value for eight IEW clock cycles. This is reserved for future use.

The Ingress Flow Control samples RES_Data during the midpoint of its eight IEW clock cycles.

The RES_Sync signal is driven to its active state during the Reserved cycle of the last time slot. The RES_Sync signal has a period equal to 512 IEW clock periods.

6.3.2.2 Configuration

The RES Bus is activated by enabling the ingress and egress functions of the internal RES data logic via the RES_Bus_Configuration_En register. Three bits in this register enable different functions:

- The I_RES_Data_En bit enables the ingress logic to capture the RES_Data.
- The E_RES_Data_En bit enables the egress logic to send its status on RES_Data.
- The E_RES_Sync_En bit enables the NP2G to drive RES_Sync.

6.3.3 Hardware Function

6.3.3.1 Exponentially Weighted Moving Average

The hardware generates exponentially weighted moving average (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 $K \in \{1, 1/2, 1/4, 1/8\}$.

6.3.3.2 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 TH_PQ+FQ, 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 TH_PQ+FQ or P0 Twin Count Threshold is exceeded, then the packet is discarded. The picocode must set up a counter block 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 zero or is less than the random number, the packet is discarded. The picocode must set up a counter block 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 TH_PQ+FQ exceeded)

6.4 The Egress Scheduler

The Egress Scheduler provides shaping functions in the network processor. The Egress scheduler manages bandwidth on a per frame basis by determining the bandwidth a frame requires (that is, the number of bytes to be transmitted) and comparing this against the bandwidth permitted by the configuration of the frame's flow queue. The bandwidth used by the first frame affects when the Scheduler permits the transmission of the second frame of a flow queue. The Egress Scheduler characterizes flow queues with the parameters listed in *Table 6-2*. *Table 6-3* lists valid combinations of the parameters described in *Table 6-2*.

Table 6-2. Flow Queue Parameters

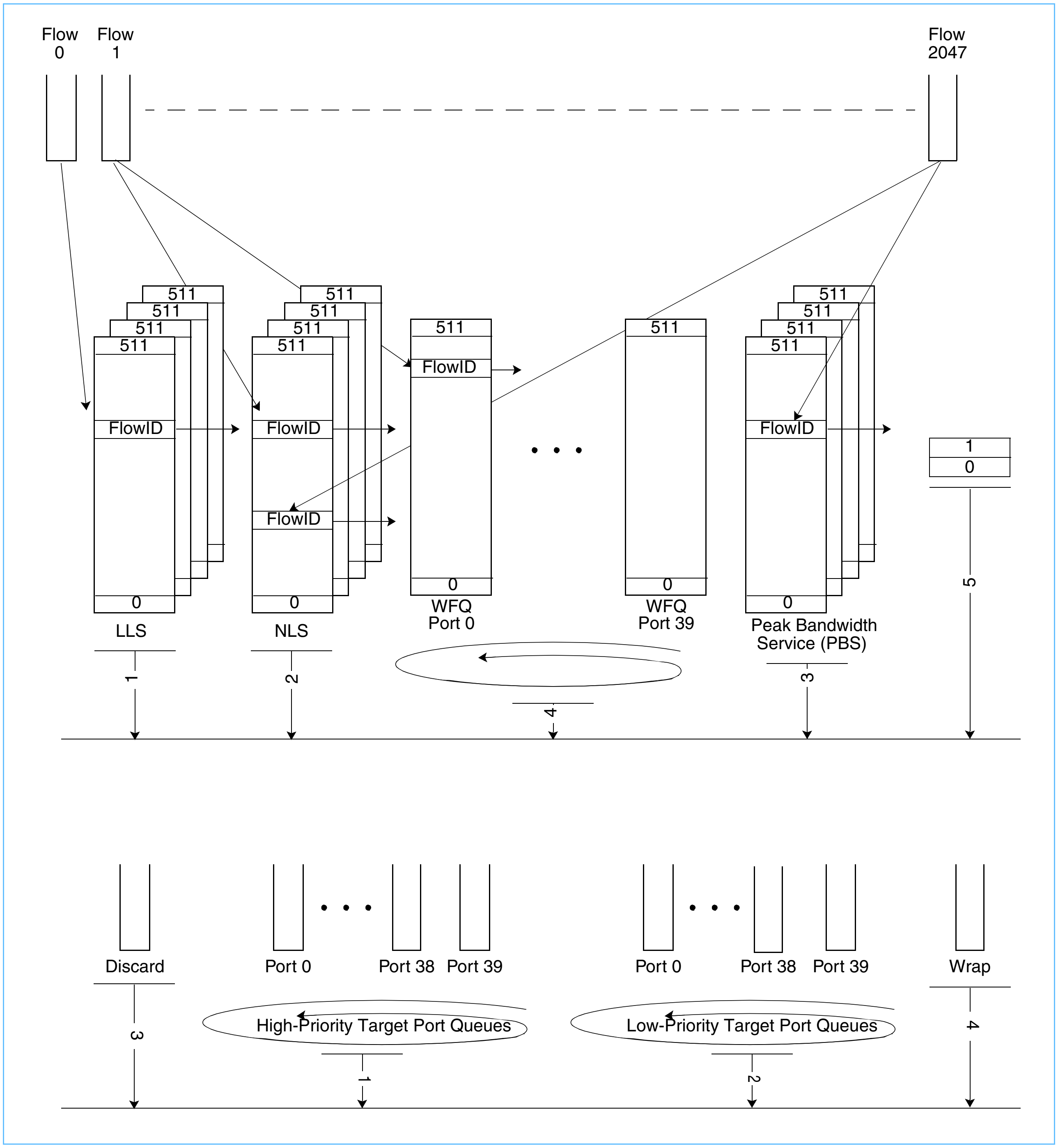
Parameter	Description
Low -latency sustainable bandwidth (LLS)	Provides guaranteed bandwidth with qualitative latency reduction. LLS has higher service priority than NLS. Flow queues connected to LLS have better latency characteristics than flow queues connected to NLS.
Normal-latency sustainable bandwidth (NLS)	Provides guaranteed bandwidth.
Peak bandwidth service (PBS)	Provides additional bandwidth on a best-effort basis.
Queue Weight	Allows the scheduler to assign available (that is, 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.

Table 6-3. 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

Figure 6-5 on page 153 presents a graphical representation of the Egress Scheduler.

Figure 6-5. The Egress Scheduler



6.4.1 Egress Scheduler Components

The Egress Scheduler consists of the following components:

- Scheduling calendars
- 2047 flow queues (flow QCB addresses 1-2047) and 2047 associated scheduler control blocks (SCBs)
- Target port queues
- Discard Queue
- Wrap Queue

6.4.1.1 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 this register field is set to 0, a scheduler_tick is 150 ns. When the field is 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 service (PBS).

Weighted Fair Queuing Calendars

The WFQ calendars allocate available bandwidth to competing flows on a per-port basis. Available bandwidth is the bandwidth left over after the flows in the time-based calendars get their bandwidth. A WFQ calendar is selected for service only when no service is required by the time-based 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.

Calendar Type Selection Algorithm

Selection among calendar types occurs each scheduler_tick based on the following priority list:

1. Time-based LLS
2. Time-based NLS
3. Time-based PBS
4. WFQ
5. Wrap

6.4.1.2 Flow Queues

There are 2047 flow queues (flow queue control block (QCB) addresses 1-2047) and scheduler control blocks. A flow QCB contains information about a single flow, as well as information that must be configured before the flow QCB can be used.

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 the SSR from typical bandwidth specifications to scheduler step units (SSD field in the QCB):

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 an SSR 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. Once the maximum frame size is known, SSD_{MAX} can be bound by:

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

For best scheduler accuracy, the smallest value for SSD.E should be specified. In order to accomplish this, $SSD.V_{MIN}$ that shall be specified is:

SSD.E	$SSD.V_{MIN}$	Comments
0	10	Allows 2.4 Gbps flow specification
1..3	32	

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

Configuring Flow QCB Flow Control Thresholds

When the number of bytes enqueued in the flow queue exceeds the Flow QCB Threshold (TH), the flow queue is 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 . A twin buffer contains 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 field in the QCB):

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 field in the QCB):

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

The flow QCB PSD field is expressed 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.

For best scheduler accuracy, the smallest value for PSD.E should be specified. In order to accomplish this, PSD.V_{MIN} that shall be specified is:

PSD.E	PSD.V_{MIN}
0	10
1..3	32

Target Port

The destination target port (TP) ID.

Target Port Priority

The Target Port Priority (P) field selects either the low-latency sustainable bandwidth (LLS) or normal-latency sustainable bandwidth (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 TP queue, the P selects the correct queue. This field is also used in flow control.

Transmit Probability

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

6.4.1.3 Target Port Queues

There are 46 target port queues, including 21 target ports, a discard port, and a wrap port, each with two priorities. 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 absolute priority selection services target port queues. The round-robin selects among the 21 media ports (target port IDs 0, 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, and 39) within each priority class (high and low, as shown in *Figure 6-5* on page 153). The absolute priority makes a secondary selection based on the following priority list:

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

The following information must be configured for each target port queue.

Port Queue Threshold (Th_PQ)

When the number of bytes enqueued in the target port queue exceeds the port queue threshold, the corresponding weighted fair queueing (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 low-latency sustainable bandwidth (LLS) and normal-latency sustainable bandwidth (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, use a value larger than the maximum transmission unit (MTU) of the target port.

Port Queue + Flow Queue Threshold (Th_PQ+FQ)

This is the 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)

6.4.2 Configuring Flow Queues

Table 6-4 on page 158 illustrates how to configure a flow QCB using the same set of Scheduler parameter combinations found in *Table 6-3: Valid Combinations of Scheduler Parameters* on page 152.

Table 6-4. 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	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

6.4.2.1 Additional Configuration Notes

- 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. The control access bus (CAB) can examine all target port queue counts; when all counts are zero, the Scheduler is drained.
- A flow queue control block (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).
- Two flow QCBs must be defined for wrap traffic. One 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 not be zero (QD ≠ 0).

6.4.3 Scheduler Accuracy and Capacity

Scheduler accuracy and capacity are affected by the 11/10 cycle select (bit 6) and the DS_D4_Refresh_Rate (bit 13) fields of the DRAM Parameter register (see *Section 13.1.2 DRAM Parameter Register (DRAM_Parm)* on page 434). These limit the number of packets per second that can be processed by the Scheduler. When this limit is exceeded, the packets are delayed, the accuracy of the scheduler is affected, and best effort traffic might not be serviced.

The 10-cycle mode corresponds to a tick rate of 150 ns and 11-cycle mode corresponds to a tick rate of 165 ns. The normal refresh rate corresponds to a refresh overhead of 1%; double refresh rate has a 2% overhead.

The number of packets per second that can be scheduled accurately is determined by the following equation:

packets per second = (1 - refresh_overhead) / tick_rate (seconds)

For example, if tick_rate = 165 ns and the refresh_overhead is 1%, the Scheduler will accurately schedule 6 million packets per second (Mpps).

For a given media rate, the minimum sustainable packet size for the Scheduler is determined by the following equation:

> bytes per packet = (1 / (1 - refresh_overhead)) * tick_rate (seconds) / packet * Media_rate (bits per second) * (1 byte / 8 bits)

Using the tick_rate and refresh_overhead from the previous equation and a Media_rate of 2.48 gigabits per second (Gbps), the minimum sustainable packet size is 51.7 bytes.

For a given minimum packet length, the maximum bandwidth that can be successfully scheduled is determined by the following equation:

> Maximum bandwidth scheduled (bits per second) = packet_length (bytes) * (8 bits/ 1 byte) * (1 / tick_rate (seconds)) * (1 - refresh_overhead)

Using the tick_rate and refresh_overhead from the previous equation and a packet length of 48 bytes, the maximum bandwidth that can be successfully scheduled is 2.3 Gbps.

7. Embedded Processor Complex

7.1 Overview

The Embedded Processor Complex (EPC) performs all processing functions for the NP2G. It provides and controls the programmability of the NP2G. In general, the EPC accepts data for processing from both the Ingress and Egress Enqueuer / Dequeuer / Schedulers. The EPC, under picocode control and with hardware-assisted coprocessors, determines what forwarding action is to be taken on the data. The data may be forwarded to its final destination, or may be discarded.

The EPC consists of the following components, as illustrated in *Figure 7-1* on page 164:

- Six Dyadic Protocol Processor Units (DPPU)

  Each DPPU consists of two Core Language Processors (CLPs), 10 shared coprocessors, one coprocessor data bus, one coprocessor command bus, and a shared memory pool.

  Each CLP contains one arithmetic and logic unit (ALU) and supports two picocode threads, so each DPPU has four threads (see *Section 7.1.1 Thread Types* on page 165 for more information). Although there are 24 independent threads, each CLP can execute the command of only one of its picocode threads, so at any instant only 12 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.

- Interrupts and Timers

  The NP2G has four Interrupt Vectors. Each interrupt can be configured to initiate a dispatch to occur to one of the threads for processing.

  The device also has 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 32 K instructions. The memory is 8-way interleaved, with each interleave providing four instruction words per access.

- 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 Tree Search Engine (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 parses 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 CLP.

  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 enables the GFH thread to control each thread on the device 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 429 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. The two algorithm pairs are “Single Rate Three Color Marker” and “Two Rate Three Color Marker,” both of which can be 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

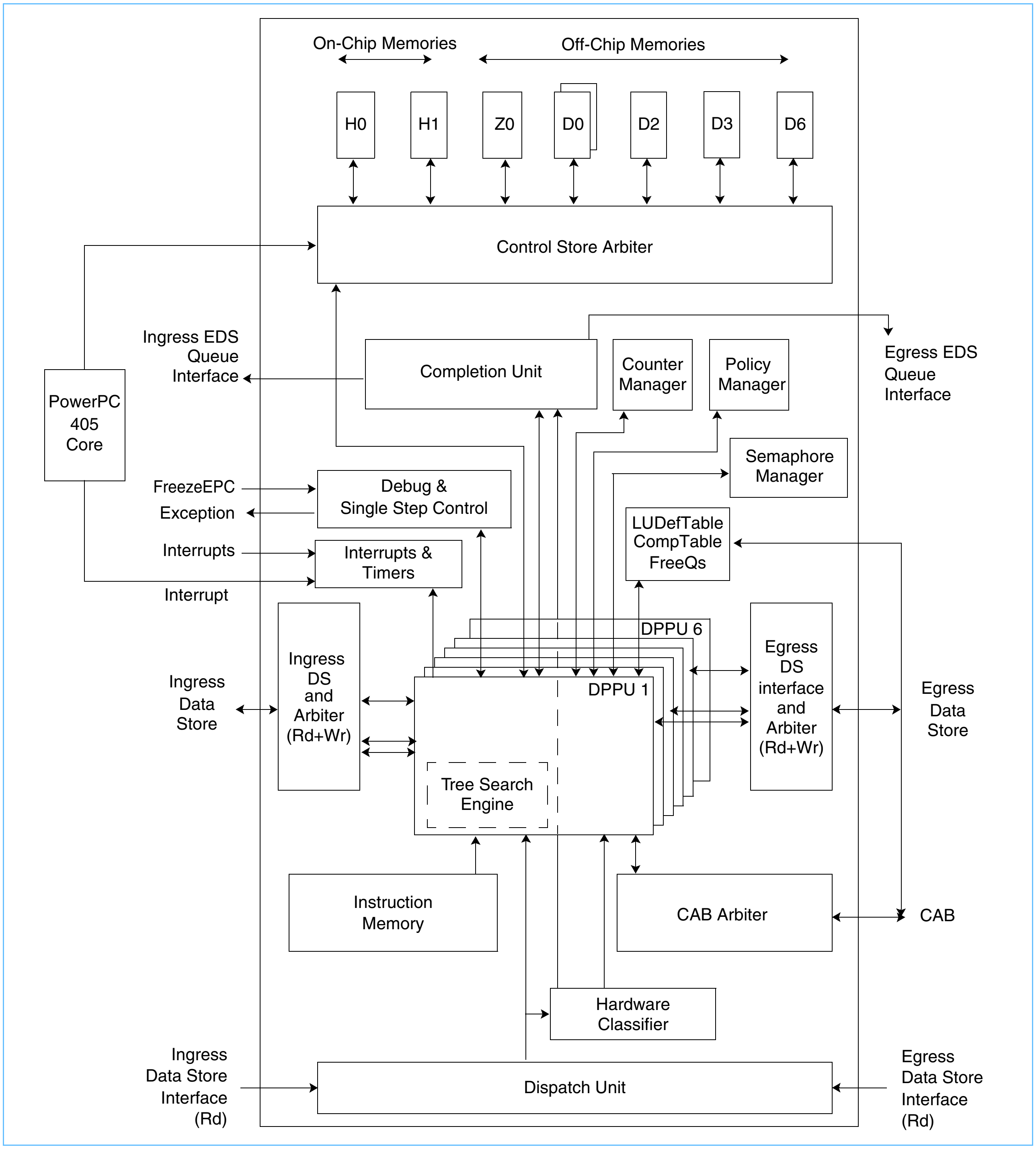
  The Counter Manager is a hardware assist engine used by the EPC to manage counters defined by the picocode for statistics, flow control, and policy management. The Counter Manager is responsible for counter updates, reads, clears, and writes. The Counter Manager's interface to the Counter coprocessor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

- Semaphore Manager

  The Semaphore Manager assists in controlling access to shared resources, such as tables and control structures, through the use of semaphores. It grants semaphores either in dispatch order (ordered semaphores) or in request order (unordered semaphores).

- LuDefTable, CompTable, and Free Queues are tables and queues for use by the tree search engine. For more information, see *Section 8.2.5.1 The LUDefTable* on page 307.

Figure 7-1. Embedded Processor Complex Block Diagram



7.1.1 Thread Types

The EPC has 24 threads that can simultaneously process 24 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. Each CLP within a DPPU can run two threads, making four threads per DPPU, or 24 total. The first DPPU contains the GFH, GTH, and the PPC threads and the other seven DPPUs contain the GDH threads. These five types of threads are described in the following list:

- General Data Handler (GDH)

  There are 20 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 device 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 381* 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 381* for more information.

7.2 Dyadic Protocol Processor Unit (DPPU)

This section describes the basic functionality of the DPPU. Two aspects of the DPPU are discussed in detail in separate sections: *Section 7.3 beginning on page 173* discusses the operational codes (opcodes) for the Core Language Processors (CLPs), and *Section 7.4 beginning on page 202* cover's the DPPU's coprocessors.

Each DPPU consists of the following functional blocks, as illustrated in *Figure 7-2*.

Two Core Language Processors (CLPs)

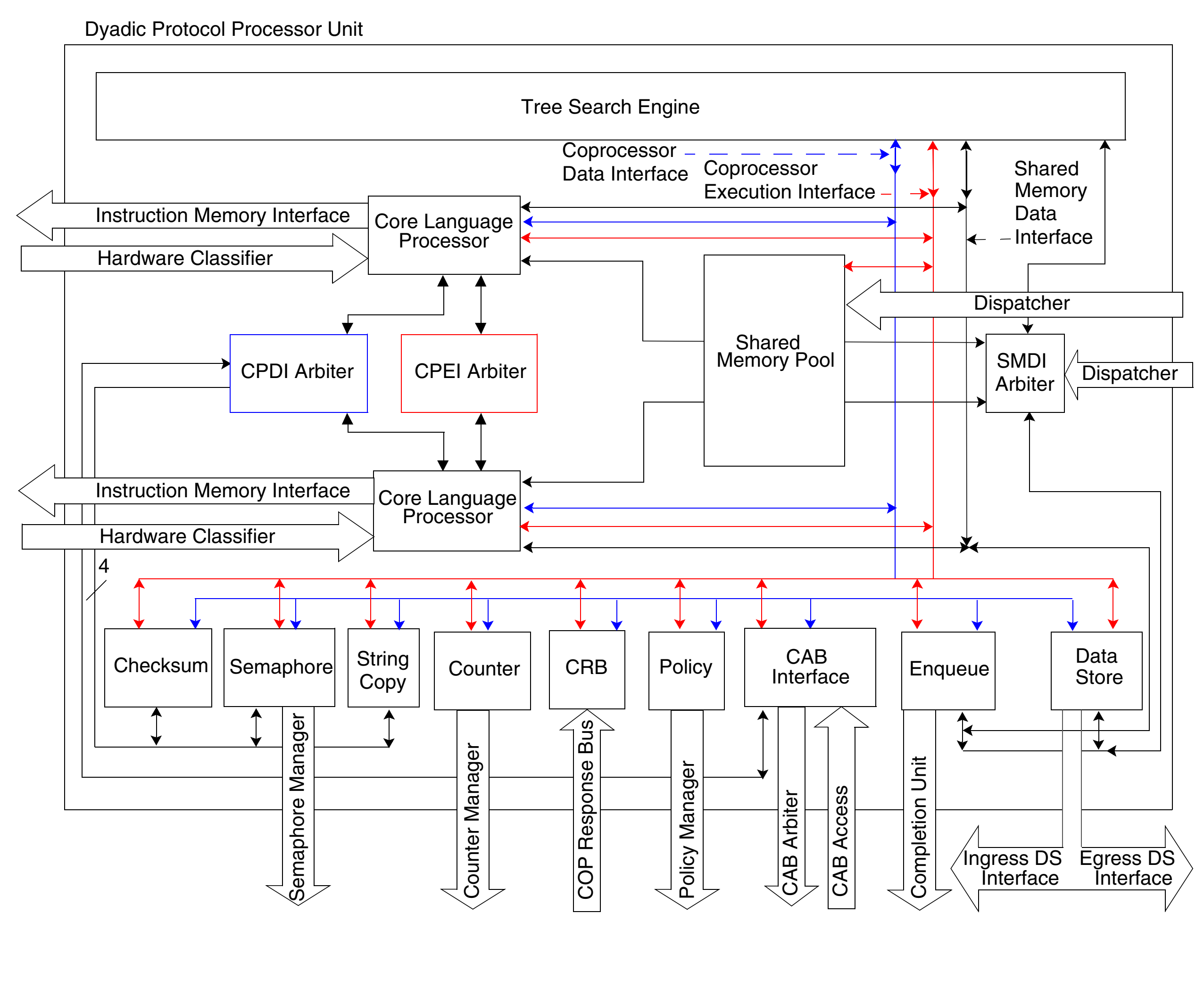
- Ten coprocessors
- A coprocessor data bus
- A coprocessor command bus (the Coprocessor Databus and Coprocessor Execution interfaces)
- A 4-KB shared memory pool (1 KB per thread)

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.

Each DPPU interfaces with the following functional blocks of the EPC:

- Instruction Memory
- Dispatch Unit
- Control Store Arbiter (CSA)
- Completion Unit (CU)
- Hardware Classifier
- Interface and arbiter to the Ingress and Egress Data Stores
- Control Access Bus (CAB) Arbiter
- Debugging facilities
- Counter Manager
- Policy Manager
- LuDefTable Queue
- Comp Free Queue
- Semaphore Manager

Figure 7-2. Dyadic Protocol Processor Unit Functional Blocks



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 (GPRs) per thread. (For more information, see *Table 7-1: Core Language Processor Address Map on page 169*.)
- 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, which improves performance and reduces 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 the following information:
  - Interrupt vectors
  - Timestamps
  - Output of a pseudo random number generator
  - Picoprocessor status
  - Work queue status (such as the ingress and egress data queues)
  - Configurable identifiers

  For more information, see *Table 7-1: Core Language Processor Address Map on page 169*.

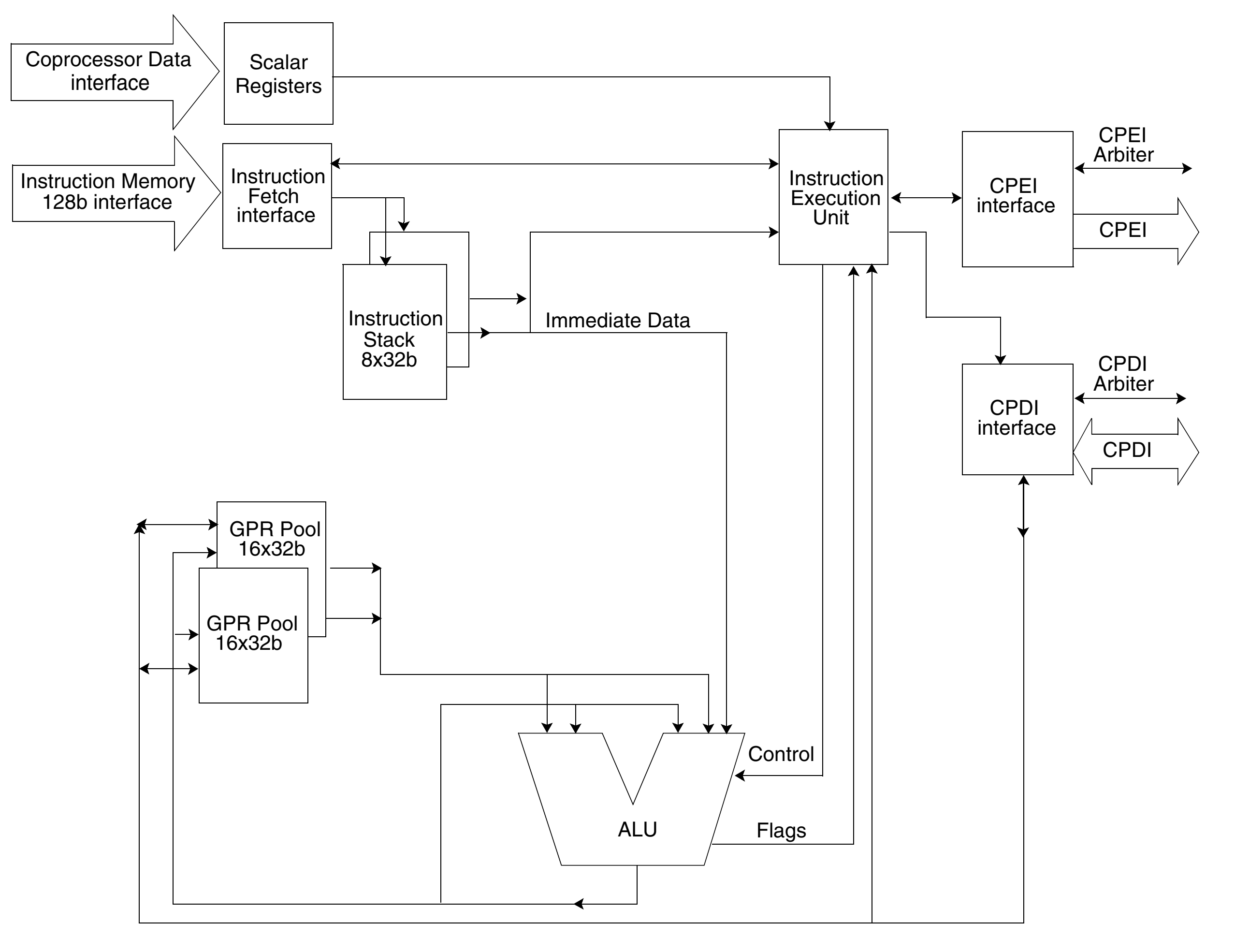
- 16-word instruction prefetch shared by each thread

- Instruction Execution unit that executes branch instructions, instruction fetch, and coprocessor access

- Coprocessor Data Interface (CPDI) with the following features:
  - 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
  - Quadword reset to zero of coprocessor arrays

- Coprocessor Execution Interface (CPEI) with the following features:
  - Synchronous or asynchronous coprocessor operation
  - Multiple coprocessor synchronization
  - Synchronization and Branch-on-Coprocessor Return Code

Figure 7-3. Core Language Processor



7.2.1.1 Core Language Processor Address Map

Table 7-1. Core Language Processor Address Map (Page 1 of 3)

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

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

Table 7-1. Core Language Processor Address Map (Page 2 of 3)

Name	Register (Array)[1] Number	Size (Bits)	Access	Description
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)
Stack_link_ptr	x‘06’	4	R	The current pointer value into the CLP link stack.
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).
Reserved	x‘89’	6	R	Reserved.
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
SW_Defined_D	x‘8D’	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
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

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

Table 7-1. Core Language Processor Address Map (Page 3 of 3)

Name	Register (Array)[1] Number	Size (Bits)	Access	Description
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 CLP Opcode Formats

The core instructions (opcodes) of the CLP, their formats, and their definitions are discussed in detail in *Section 7.3 beginning on page 173*.

7.2.3 DPPU Coprocessors

All data processing occurs in the ten DPPU coprocessors. They are discussed in detail in *Section 7.4 beginning on page 202*.

7.2.4 Shared Memory Pool

The 4-Kb shared memory pool is used by all threads running in the DPPU. Each thread uses 1 Kb and is subdivided into the following areas:

Table 7-2. 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 CLP Opcode Formats

This section describes the core instructions (opcodes) of the CLP, their formats, and their definitions. CLP opcodes fall into four categories: control opcodes, ALU opcodes, data movement opcodes, and coprocessor execution opcodes. Shaded areas of the opcode show which bits uniquely identify the opcode.

7.3.1 Control Opcodes

Most control opcodes have a condition field (Cond) that indicates under what condition of the ALU flags Z (zero), C (carry), N (negative), and V (overflow) this command should be executed. The CLP supports 15 different signed and unsigned conditions.

Table 7-3. Condition Codes (Cond Field)

Value	ALU Flag Comparison	Meaning
0000	Z = 1	equal or zero
0001	Z = 0	not equal or not zero
0010	C = 1	carry set
0011	C = 1 AND Z = 0	unsigned higher
0100	C = 0 OR Z = 1	unsigned lower or equal
0101	C = 0	unsigned lower
0110	-	Reserved
0111	Don’t Care	Always
1000	N = 0	signed positive
1001	S = 1	signed negative
1010	N = V	signed greater or equal
1011	Z = 0 AND N = V	signed greater than
1100	Z = 1 OR (N/ = V)	signed less than or equal
1101	N/ = V	signed less than
1110	V = 1	overflow
1111	V = 0	no overflow

Type	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
nop	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
exit	0	0	0	1	0	0	0	0	Cond				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
test and branch	0	0	0	1	0	1	c	bp					R				disp16															
branch and Link	0	0	0	1	1	h	1	1	Cond				Rt				target16															
return	0	0	0	1	1	0	0	0	Cond				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
branch register	0	0	0	1	1	h	0	0	Cond				Rt				0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
branch pc relative	0	0	0	1	1	0	1	0	Cond				0	0	0	0	disp16															
branch reg+off	0	0	0	1	1	0	0	1	Cond				Rt				target16															

7.3.1.1 Nop Opcode

The nop opcode executes one cycle of time and does not change any state within the processor.

7.3.1.2 Exit Opcode

The exit opcode terminates the current instruction stream. The CLP will be put into an idle state and made available for a new dispatch. The exit command is executed conditionally, and if the condition is not true, it will be equivalent to a nop opcode.

pseudocode:

```
IF (cond) THEN
    clp_state_machine <- idle
ELSE
     nop
END IF
```

7.3.1.3 Test and Branch Opcode

The test and branch opcode tests a single bit within a GPR register. If the desired condition is met, the thread will branch. The R field is the GPR register to be tested. The bp field represents the bit position within the GPR to be tested. The c field represents the test condition: if c = 1, the branch is taken if the bit tested is a ‘1’; if c = 0, the branch is taken if the bit tested is a ‘0’. The branch target ID is calculated by using the disp16 field as a displacement from the program counter (PC). This command does not set the ALU flags.

pseudocode:

```
IF R(bp)=c THEN
     PC <- PC + disp16
    ELSE
     nop
END IF
```

7.3.1.4 Branch and Link Opcode

The branch and link opcode performs a conditional branch, adds one to the value of the current program counter, and places it onto the program stack. Opcode fields Rt, h, and target16 determine the destination of the branch. If Rt is in the range of 1 - 15, it is the word address of a GPR register, the contents of which are used as the base of the branch destination. If Rt is 0, the base of the branch destination will be 0x‘0000’. The h field indicates which half of the GPR register is used as the base address. An h = 0 indicates the even half of the GPR register (high half of the GPR) and h = 1 indicates the odd half. The target16 field is added to the base to form the complete branch destination. If the LinkPtr register indicates that the Branch and Link will overflow the 16-entry program stack, a stack error occurs.

pseudocode:

```
IF (cond) THEN

     -- put IA+1 onto the program stack
     IF (LinkPtr=EndOfStack) THEN
          StackErr <- 1
     ELSE
           ProgramStack(LinkPtr+1) <- PC + 1
     END IF

     -- load the IA with the branch target
     IF (Rt=0) THEN
          PC <- target16
     ELSIF (h=0) THEN
          PC <- GPR(Rt)(31:16) + target16
     ELSE
          PC <- GPR(Rt)(15:0) + target16
     END IF

ELSE
     nop
END IF
```

7.3.1.5 Return Opcode

The return opcode performs a conditional branch with the branch destination being the top of the program stack. If the LinkPtr register indicates that the stack is empty, a stack error occurs.

pseudocode:

```
IF (cond) THEN
     IF (LinkPtr=EmptyStack) THEN
          StackErr <- 1
     ELSE
           PC <- ProgramStack(LinkPtr)
     END IF
ELSE
   nop
END IF
```

7.3.1.6 Branch Register Opcode

The branch register opcode performs a conditional branch. Opcode fields Rt and h determine the destination of the branch. The Rt field is the word address of a GPR register of which the contents will be used as the branch destination. The h field indicates which half of the GPR register will be used. An h = 0 indicates the even half of the GPR register (high half of the GPR), and h = 1 indicates the odd half.

pseudocode:

```
IF (cond) THEN
     IF (h=0) THEN
          PC <- GPR(Rt)(31:16) + target16
     ELSE
          PC <- GPR(Rt)(15:0) + target16
     END IF
ELSE
     nop
END IF
```

7.3.1.7 Branch PC Relative Opcode

The branch PC relative opcode performs a conditional branch. Opcode fields PC and disp16 determine the destination of the branch. The contents of the PC field are used as the base of the branch destination. The disp16 field will be added to the base to form the complete branch destination.

pseudocode:

```
IF (cond) THEN
     PC <- PC + disp16
    ELSE
     nop
END IF
```

7.3.1.8 Branch Reg+Off Opcode

The branch register plus offset opcode will perform a conditional branch. The Opcode fields Rt and target16 determine the destination of the branch. If Rt is in the range of 1 - 15, it is the word address of a GPR register of which the contents of the high halfword (or even half) are used as the base of the branch destination. If Rt is 0, the base of the branch destination is x‘0000’. The target16 field is added to the base to form the complete branch destination.

pseudocode:

```
IF (cond) THEN
     IF (Rt=0) THEN
          PC <- target16
     ELSE
          PC <- GPR(Rt)(31:16) + target16
     END IF
ELSE
     nop
END IF
```

7.3.2 Data Movement Opcodes

The data movement opcodes are used to transfer data to and from the coprocessor's scalar registers and arrays. The opcodes support 23 options of direction, size, extension, and fill, represented by the ot5 field. Data movement opcodes access the processor data interface (PDI).

Figure 7-4. ot5 Field Definition: Loading Halfword/Word GPRs from a Halfword/Word Array

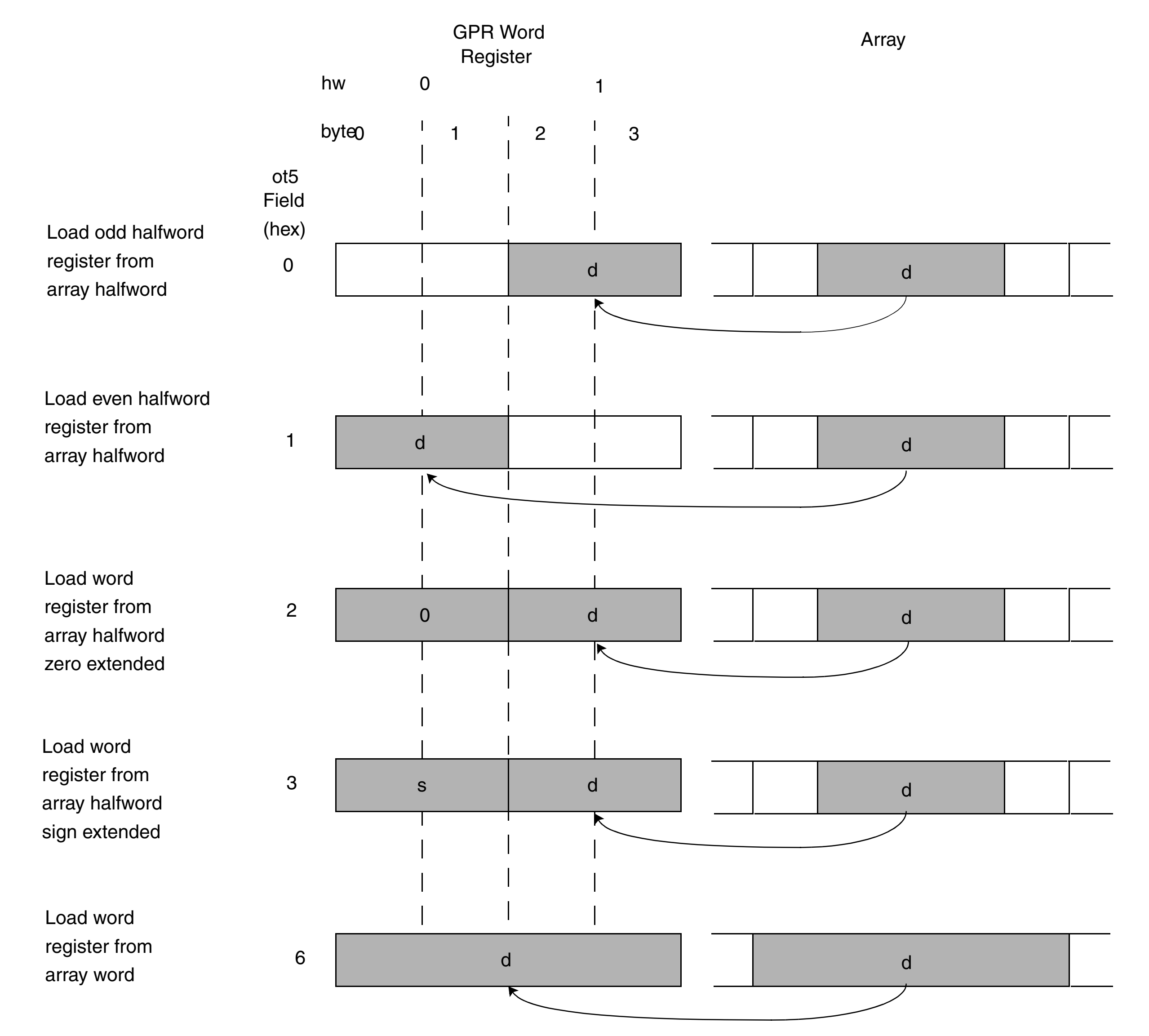


Figure 7-5. ot5 Field Definition: Loading GPR Byte From Array Byte

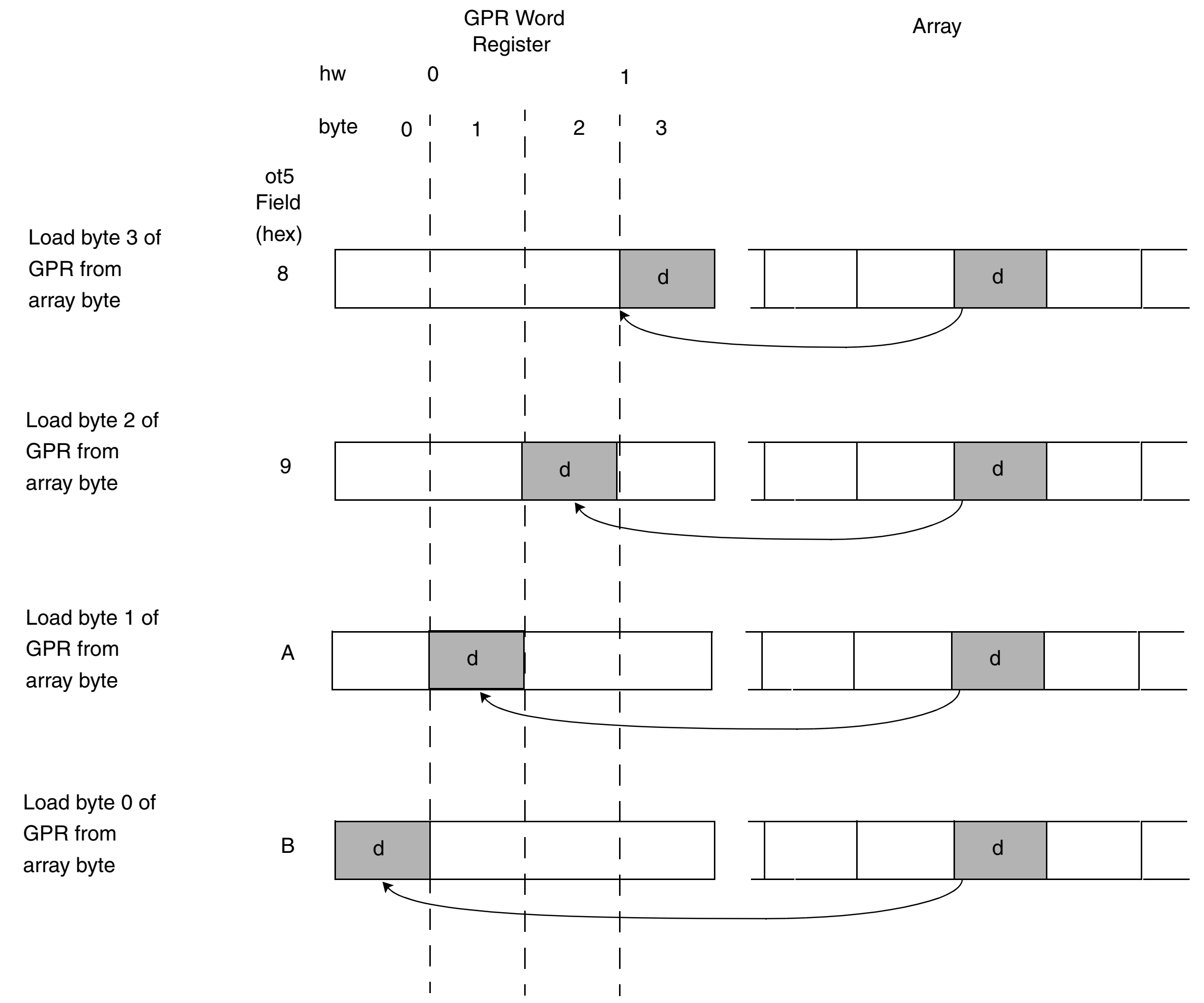

GPR Word Register
Array
hw
0
1
byte
0
1
2
3
ot5 Field (hex)
Load byte 3 of GPR from array byte
8
d
d
Load byte 2 of GPR from array byte
9
d
d
Load byte 1 of GPR from array byte
A
d
d
Load byte 0 of GPR from array byte
B
d
d

Figure 7-6. ot5 Field Definition: Loading GPR Halfword/Word from Array Byte

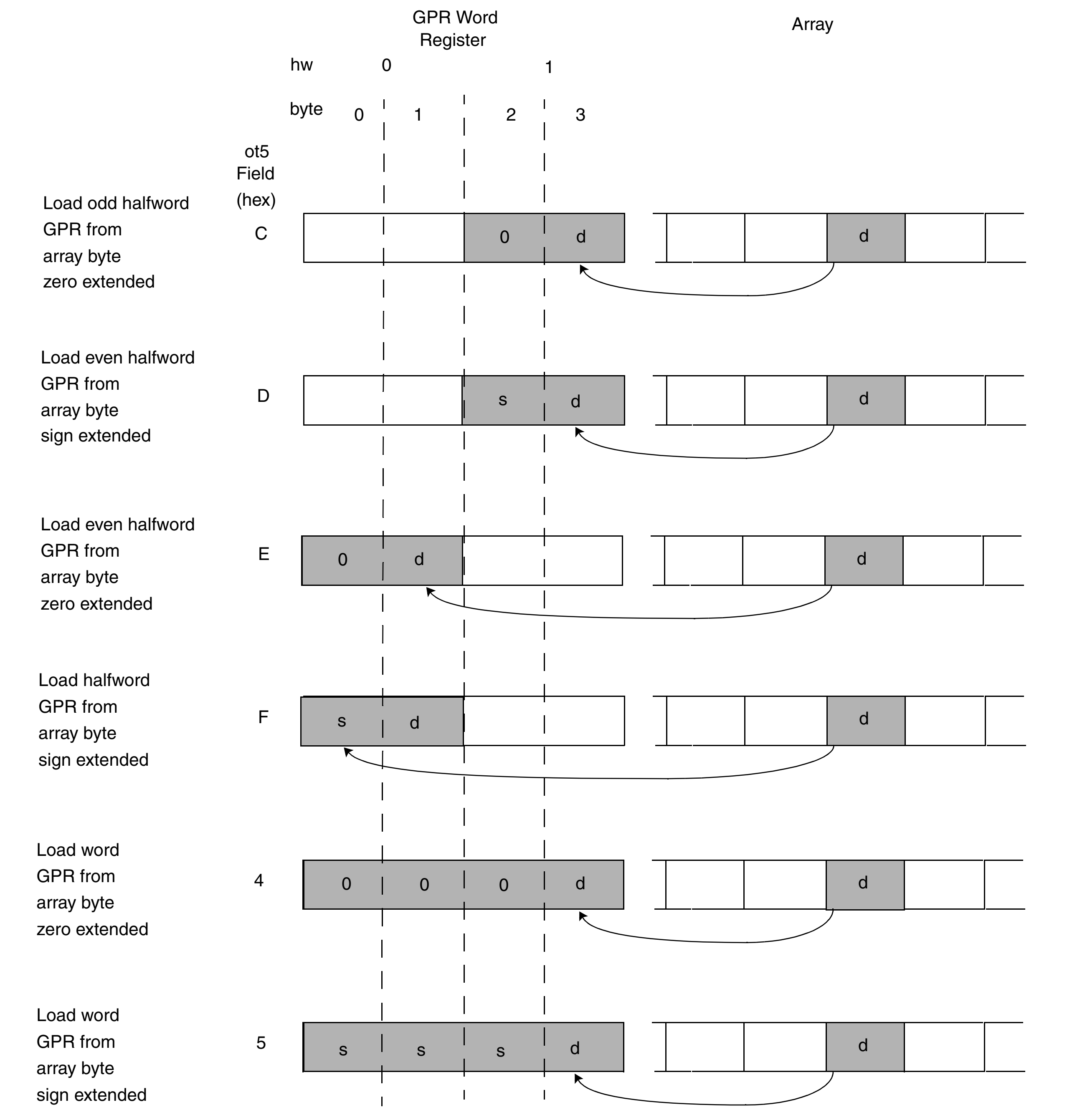
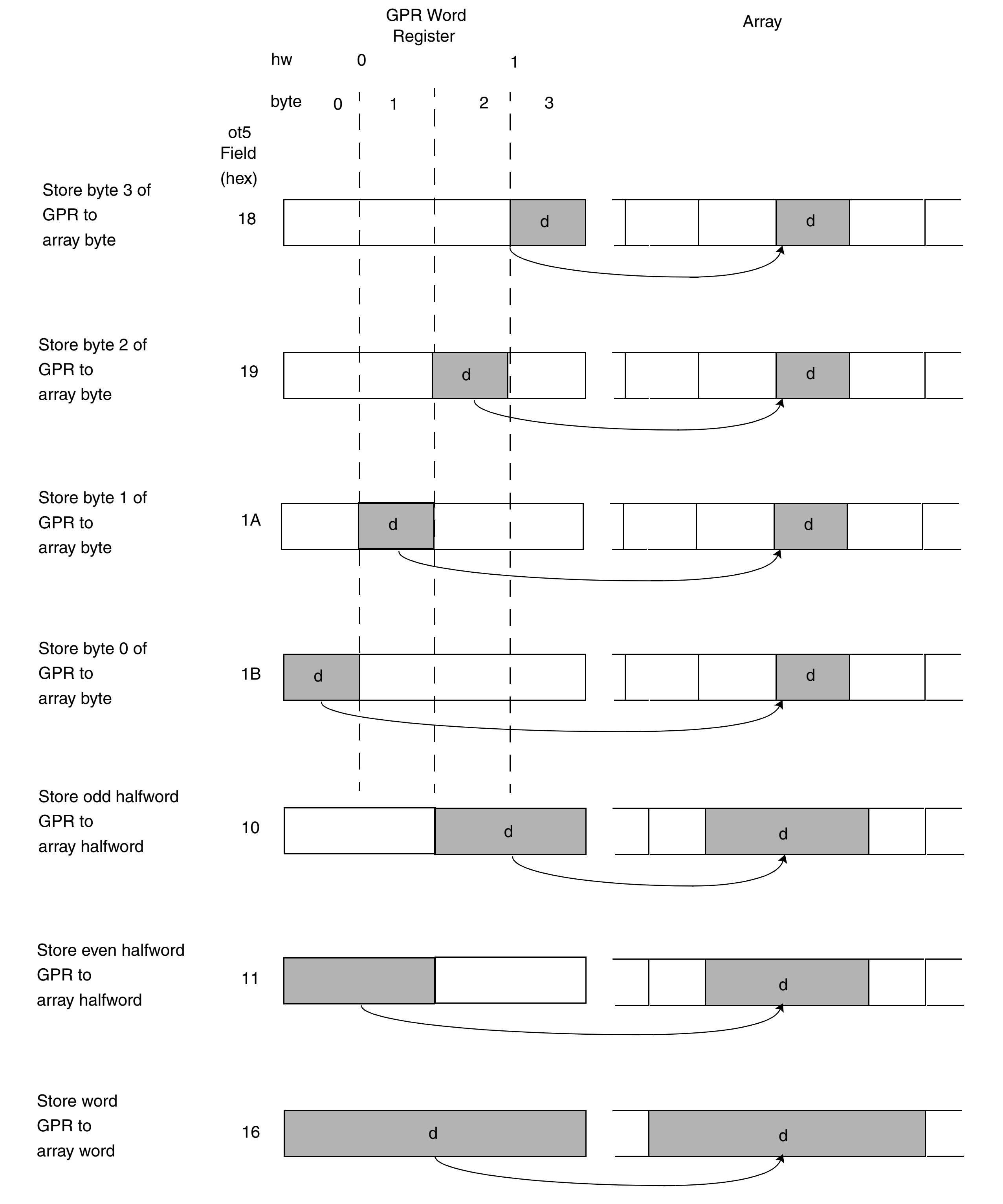

GPR Word Register
Array
hw
0
1
byte
0
1
2
3
ot5 Field (hex)
Load odd halfword GPR from array byte zero extended
C
0
d
d
Load even halfword GPR from array byte sign extended
D
s
d
d
Load even halfword GPR from array byte zero extended
E
0
d
d
Load halfword GPR from array byte sign extended
F
s
d
d
Load word GPR from array byte zero extended
4
0
0
0
d
d
Load word GPR from array byte sign extended
5
s
s
s
d
d

Figure 7-7. ot5 FIeld Definition: Store GPR Byte/Halfword/Word to Array Byte/Halfword/Word



Type	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
memory indirect	0	0	1	ot5					Cond				R				x	Ra			C#				Ca		off6					
memory add indirect	0	1	0	ot5					Cond				R				x	Ra			0	0	0	0	off8							
memory direct	0	1	1	ot5					Cond				R				x	0	off2		C#				Ca		off6					
scalar access	0	1	1	ot5					Cond				R				0	1	0	0	C#				Cr							
scalar immed	1	1	0	1	C#				Cr								Imm16															
transfer QW	1	1	0	0	C#d				Cad		0	0	QWoffd				0	0	0	0	C#s				Cas		0	1	QWoffs			
zero array	1	1	0	0	0	0	size		Cond				Boffset				x	0	0	0	C#				Ca		1	1	QWoffs			

7.3.2.1 Memory Indirect Opcode

The memory indirect opcode transfers data between a GPR and a coprocessor array via a logical address in which the base offset into the array is contained in a GPR. The logical address consists of a coprocessor number, coprocessor array, base address, and an offset. The C# field indicates the coprocessor which will be accessed. The Ca field indicates which array of the coprocessor will be accessed. The offset into the array is the contents of GPR, indicated by Ra as a base plus the six bits of immediate offset, off6. Ra is a half-word address in the range of 0 - 7. The x indicates whether or not the transaction will be in cell header skip mode. The memory indirect opcode is executed conditionally if the ALU flags meet the condition represented by the Cond field. The word address of the GPR register used to transfer or receive data from the array is indicated by the field R. The actual direction, size, extension format, and location of GPR bytes affected are indicated by the ot5 field.

pseudocode:

```
IF (cond) THEN
    addr.coprocessor number <= C#
    addr.coprocessor array     <= Ca
    addr.array offset                <= GPR(Ra)+off6
    addr.x_mode                     <= x

    IF (ot5 = load) THEN
       GPR(R,ot5) <= array(addr)
    ELSE
       array(addr) <= GPR(R,ot5)
    END IF;

END IF
```

7.3.2.2 Memory Address Indirect Opcode

The memory address indirect opcode transfers data between a GPR and a coprocessor data entity (scalar or array) by mapping the coprocessor logical address into the base address held in the GPR indicated by Ra. Since not all of the coprocessors have arrays, maximum size arrays, or the maximum number of scalars, the address map will have many gaps. Data access via this method is the same as access via the more logic-based method. The final address is formed by the contents of GPR indicated by Ra plus off8 (eight bits). Ra is a half-word address in the range of 0 - 7. The x indicates whether or not the transaction will be in cell header skip mode.

The memory address indirect opcode is executed conditionally if the ALU flags meet the condition represented by the Cond field. The word address of the GPR register used to transfer or receive data from the array is indicated by the field R. The actual direction, size, extension format, and location of GPR bytes affected are indicated by the ot5 field.

pseudocode:

```
IF (cond) THEN
    address <= GPR(Ra) + off8

    addr.array_notScalar         <= address(14)
    addr.coprocessor number <= address(13:10)
    addr.coprocessor array     <= address(9:8) (if an array access)
    addr.scalar address           <= address(7:0) (if a scalar access)
    addr.array offset                 <= address(7:0) (if an array access)
    addr.x_mode                      <= x

    IF (ot5 = load) THEN
       IF (addr.array_notScalar='1') THEN
         GPR(R,ot5) <= array(addr)
       ELSE
         GPR(R,ot5) <= scalar(addr)
       END IF;
    ELSE
       IF (addr.array_notScalar='1') THEN
          array(addr) <= GPR(R,ot5)
       ELSE
         scalar(addr) <= GPR(R,ot5)
       END IF;
    END IF;
END IF
```

7.3.2.3 Memory Direct Opcode

The memory direct opcode transfers data between a GPR and a coprocessor array via a logical address that is specified in the immediate portion of the opcode. The logical address consists of a coprocessor number, coprocessor array, and an offset. The C# field indicates the coprocessor that is accessed. The Ca field indicates which array of the coprocessor is accessed. The offset is made up of eight bits: the most significant two bits are located in field off2 and the remaining six bits are in off6. The x indicates whether or not the transaction will be in cell header skip mode. The memory direct opcode is executed conditionally if the ALU flags meet the condition represented by the Cond field. The word address of the GPR register used to transfer or receive data from the array is indicated by the R field. The actual direction, size, extension format, and location of GPR bytes affected are indicated by the ot5 field.

pseudocode:

```
IF (cond) THEN
    addr.coprocessor number <= C#
    addr.coprocessor array     <= Ca
    addr.array offset                 <= off2 & off6
    addr.x_mode                      <= x

    IF (ot5 = load) THEN
       GPR(R,ot5) <= array(addr)
    ELSE
       array(addr) <= GPR(R,ot5)
    END IF;

END IF
```

7.3.2.4 Scalar Access Opcode

The scalar access opcode transfers data between a GPR and a scalar register via a logical address that consists of a coprocessor number and a scalar register number. The C# field indicates the coprocessor that is accessed. The scalar register number is indicated by the Cr field. The scalar access opcode is executed conditionally if the ALU flags meet the condition represented by the Cond field. The word address of the GPR register used to transfer or receive data from the scalar register is indicated by the R field. The actual direction, size, extension format, and location of GPR bytes affected are indicated by the ot5 field.

pseudocode:

```
IF (cond) THEN
    addr.coprocessor number <= C#
    addr.scalar number           <= Ca

    IF (ot5 = load) THEN
       GPR(R,ot5) <= scalar(addr)
    ELSE
      scalar(addr) <= GPR(R,ot5)
    END IF;
END IF
```

7.3.2.5 Scalar Immediate Opcode

The scalar immediate opcode writes immediate data to a scalar register via a logical address that is completely specified in the immediate portion of the opcode. The logical address consists of a coprocessor number and a coprocessor register number. The C# field indicates the coprocessor that is accessed. The Cr field indicates the scalar register number. The data to be written is the Imm16 field.

pseudocode:

```
addr.coprocessor number    <= C#
addr.scalar register number <= Ca
scalar(addr) <= Imm16
```

7.3.2.6 Transfer Quadword Opcode

The transfer quadword opcode transfers quadword data from one array location to another using one instruction. The source quadword is identified by the C#s (coprocessor number), Cas (source array number), and QWoffs (quadword offset into the array). The destination quadword is identified by the C#d, Cad, and QWoffd.This transfer is only valid on quadword boundaries.

pseudocode:

```
saddr.coprocessor number <= C#s
saddr.array number             <= Cas
saddr.quadword offset         <= QWoffs
daddr.coprocessor number <= C#d
daddr.array number             <= Cad
daddr.quadword offset         <= QWoffd

array(daddr) <= array(saddr)
```

7.3.2.7 Zero Array Opcode

The zero array opcode zeroes out a portion of an array with one instruction. The size of the zeroed-out portion can be a byte, halfword, word, or quadword. Quadword access must be on quadword address boundaries. Accesses to a byte, halfword, or word can begin on any byte boundary and will have the same characteristics as any GPR-based write to the array. For example, if the array is defined to wrap from the end to the beginning, the zero array command wraps from the end of the array to the beginning. The C# field indicates the coprocessor that will be accessed. The Ca field indicates which array of the coprocessor is accessed. The QWoff is the quadword offset into the array, and the Boff is the byte offset into the array. The x indicates whether or not the transaction is in cell header skip mode. The opcode is executed conditionally if the ALU flags meet the condition represented by the Cond field. The actual size of the access is defined by the size field (byte = 00, halfword = 01, word = 10, quadword = 11). For quadword accesses Boff should equal 0x0.

pseudocode:

```
IF (cond) THEN
  addr.coprocessor number <= C#
  addr.array number            <= Ca
  addr.quadword offset         <= QWoff
  addr.byte offset                  <= Boff

  IF size = 00 THEN
     array(addr) <= 0x00
  IF size = 01 THEN
     array(addr : addr+1) <= 0x0000
  IF size = 10 THEN
     array(addr : addr+3) <= 0x00000000
  IF size = 11 THEN
     array(addr : addr+15) <= 0x00000000000000000000000000000000
END IF;
```

7.3.3 Coprocessor Execution Opcodes

The coprocessor execution opcodes are used to initiate operations or synchronize operations with the coprocessors. The execution type opcodes initiate accesses on the processor execution interface (PEI). A command to a coprocessor consists of six bits of coprocessor operation and 44 bits of coprocessor arguments. The definition of the operation and argument bits is coprocessor dependent.

A coprocessor can operate synchronously or asynchronously. Synchronous operation means the execution of the thread initiating the coprocessor command stalls until the coprocessor operation is complete. Asynchronous operation means the executing thread will not stall when a coprocessor execution command is issued, but rather can continue operating on the instruction stream. It is important to re-synchronize a coprocessor command that was issued asynchronously before using resources that the coprocessor needs for execution of the command. This can be done with the “wait” coprocessor execution opcodes.

The CLP runs the instruction stream of two threads in which only one thread can actually execute in a given cycle. The CLP thread that owns priority cannot be stalled by the execution of the non-priority thread. Priority is granted to the only active thread in a CLP or to the thread that is given priority by the other thread. The coprocessor execution opcodes indicated by the P bit in the following opcode map can give up the priority status of the thread.

Type	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
execute direct	1	1	1	1	C#				p	CPop					a	0	Imm16															
execute indirect	1	1	1	1	C#				p	CPop					a	1	R				Imm12											
execute direct conditional	1	1	1	0	C#				Cond				op		0	1	Imm16															
execute indirect conditional	1	1	1	0	C#				Cond				op		1	1	R				Imm12											
wait	1	1	1	0	0	0	0	0	p	0	0	0	0	0	0	0	mask16															
wait and branch	1	1	1	0	C#				p	0	0	0	1	0	ok	0	Target16															

7.3.3.1 Execute Direct Opcode

The execute direct opcode initiates a coprocessor command in which all of the operation arguments are passed immediately to the opcode. The C# field indicates which coprocessor will execute the command. The CPopfield is the coprocessor operation field. The Imm16 field contains the operation arguments that are passed with the command. The a field indicates whether the command should be executed asynchronously. The p field indicates if the thread should give up priority.

pseudocode:

```
exe.coprocessor number     <= C#
  exe.coprocessor operation  <= CPop
  exe.coprocessor arguments <=  “0000000000000000000000000000” & Imm16

  coprocessor <= exe

  IF a=1 THEN
     PC <= PC+1
  ELSE
     PC <= stall
  END IF

  IF p=1 THEN
    PriorityOwner(other thread)<= TRUE
  ELSE
    PriorityOwner(other thread)<= PriorityOwner(other thread)
  END IF;
```

7.3.3.2 Execute Indirect Opcode

The execute indirect opcode initiates a coprocessor command in which the operation arguments are a combination of a GPR register and an immediate field. The field C# indicates which coprocessor the command is to be executed on. The field CPop is the coprocessor operation field. The R field is the GPR register to be passed as part of the operation arguments. The Imm12 field contains the immediate operation arguments that are passed. The a field indicates whether the command should be executed asynchronously. The p field indicates if the thread should give up priority.

pseudocode:

```
exe.coprocessor number     <= C#
  exe.coprocessor operation  <= CPop
  exe.coprocessor arguments <= Imm12 & GPR(R)

  coprocessor <= exe

  IF a=1 THEN
     PC <= PC+1
  ELSE
     PC <= stall
  END IF

  IF p=1 THEN
    PriorityOwner(other thread)<= TRUE
  ELSE
    PriorityOwner(other thread)<= PriorityOwner(other thread)
  END IF;
```

7.3.3.3 Execute Direct Conditional Opcode

The execute direct conditional opcode is similar to the execute direct opcode except that the execute direct conditional opcode command can be issued conditionally based on the Cond field. To make room in the opcode for the Cond field, the coprocessor opcode field (op) is shortened to two bits. Because the high order four bits of the coprocessor operation are assumed to be zeros, conditional operations are restricted to the lower four commands of a coprocessor. he command is assumed to be synchronous because the opcode does not have a bit to indicate whether it is asynchronous or synchronous. Priority cannot be released with this opcode.

pseudocode:

```
IF (Cond) THEN
     exe.coprocessor number      <= C#
     exe.coprocessor operation   <= “0000”&op
  exe.coprocessor arguments <=  “0000000000000000000000000000” & Imm16

     coprocessor <= exe
END IF
```

7.3.3.4 Execute Indirect Conditional Opcode

The execute indirect conditional opcode is similar to the execute indirect opcode except that the execute indirect command can be issued conditionally based on the Cond field. To make room in the opcode for the Cond field, the coprocessor opcode field (op) is shortened to two bits. Because the high order four bits of the coprocessor operation are assumed to be 0s, conditional operations are restricted to the lower four commands of a coprocessor. The command is assumed to be synchronous because the opcode does not have a bit to indicate whether it is asynchronous or synchronous. Priority cannot be released with this opcode.

pseudocode:

```
IF (Cond) THEN
     exe.coprocessor number      <= C#
     exe.coprocessor operation   <= “0000”&op
     exe.coprocessor arguments  <= Imm12 & GPR(R)

     coprocessor <= exe
END IF
```

7.3.3.5 Wait Opcode

The wait opcode synchronizes one or more coprocessors. The mask16 field (see the register bit list table in *Section 7.3.3* on page 187) is a bit mask (one bit per coprocessor) in which the bit number corresponds to the coprocessor number. The thread stalls until all coprocessors indicated by the mask complete their operations. Priority can be released with this command.

pseudocode:

```
IF Reduction_OR(mask16(i)=coprocessor.busy(i)) THEN
   PC <= stall
ELSE
   PC <= PC + 1
END IF

IF p=1 THEN
  PriorityOwner(other thread)<= TRUE
ELSE
  PriorityOwner(other thread)<= PriorityOwner(other thread)
END IF;
```

7.3.3.6 Wait and Branch Opcode

The wait and branch opcode synchronizes with one coprocessor and branch on its one bit OK/KO flag. This opcode causes the thread to stall until the coprocessor represented by C# is no longer busy. The OK/KO flag is then compared with the field OK. If they are equal, the thread branches to the address in the target16 field. Priority can be released with this command.

pseudocode:

```
IF coprocessor.busy(C#)=1 THEN
   PC <= stall
ELSIF coprocessor.ok(C#)=ok THEN
   PC <= target16
ELSE
   PC <= PC+1
END IF

IF p=1 THEN
  PriorityOwner(other thread)<= TRUE
ELSE
  PriorityOwner(other thread)<= PriorityOwner(other thread)
END IF;
```

7.3.4 ALU Opcodes

ALU opcodes are used to manipulate GPR registers with arithmetic and logical functions. All of these opcodes execute in a single cycle. These opcodes are also used to manipulate the ALU status flags used in condition execution of opcodes. All ALU opcodes are executed conditionally based upon the Cond field.

7.3.4.1 Arithmetic Immediate Opcode

Type	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Arithmetic Immediate	1	0	0	0	i	ot3i			Cond				R				Imm12(11:8)				AluOp				Imm12(7:0)							
Logical Immediate	1	0	0	1	i	h	Lop		Cond				R				Imm16															
Compare Immediate	1	0	1	0	0	0	ot2i		Cond				R				Imm16															
Load Immediate	1	0	1	1	ot4i				Cond				R				Imm16															
Arithmetic/ Logical Register	1	1	0	0	i	ot3r			Cond				Rd				Rs				AluOp				0	0	0	0	w	h	n	m
Count Leading Zeros	1	0	1	0	1	hd	w	i	Cond				Rd				Rs				0	0	0	0	0	0	0	0	0	h	n	0

The arithmetic immediate opcode performs an arithmetic function on a GPR register in which the second operand is a 12-bit immediate value. The word GPR operand and destination GPR are specified in the R field and the immediate operand is represented by Imm4&Imm8. The actual portion of the GPR and extension of the immediate operand are shown in the ot3i table. The arithmetic function performed is given in the AluOp field. This AluOp functions of AND, OR, XOR, TST, and COMPARE are not used with this opcode and have a different opcode when the second operand is an immediate value. Arithmetic opcodes can be performed without changing the ALU status flags if the i field is a 1.

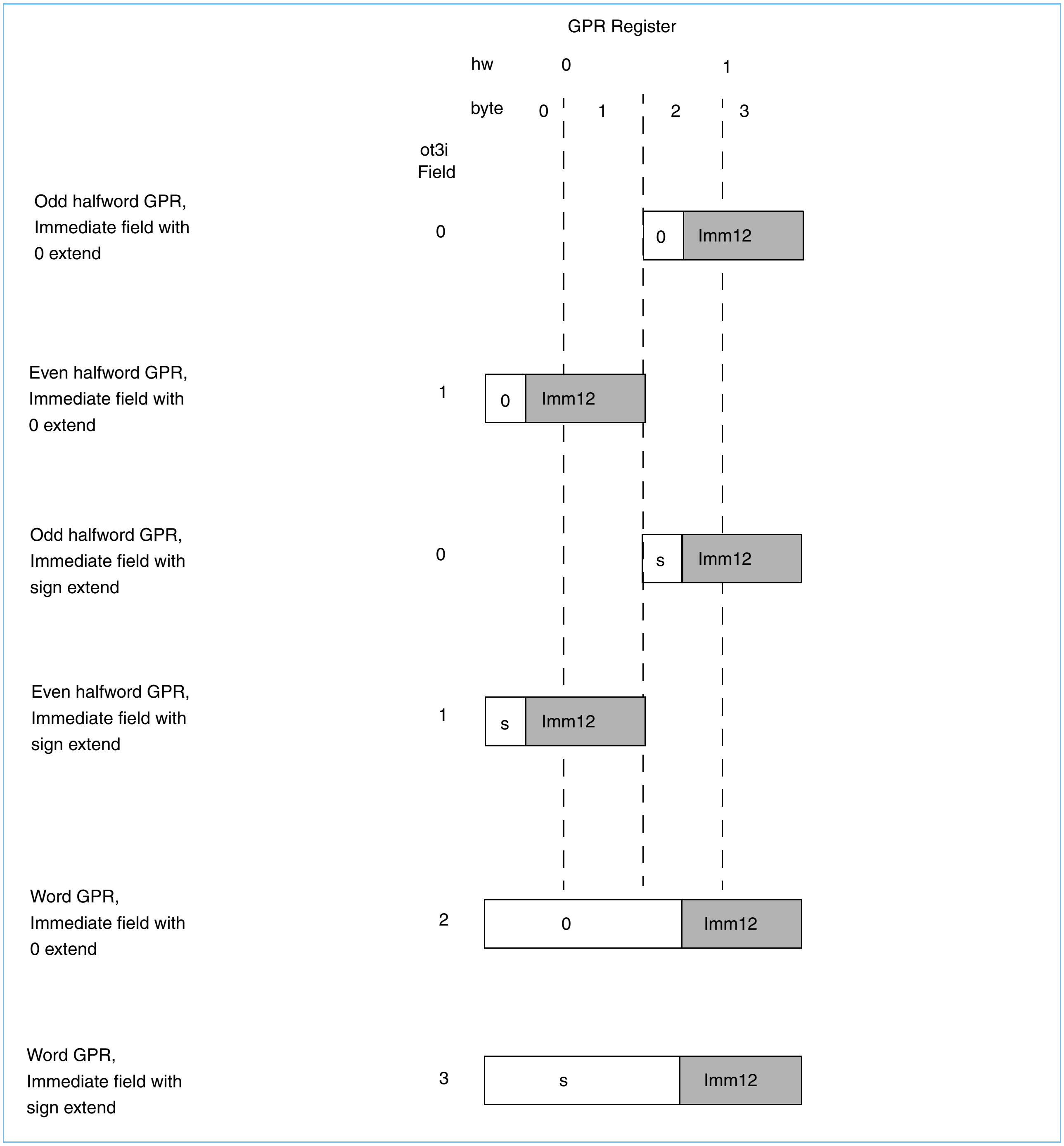
pseudocode:

```
IF (Cond) THEN
     alu.opr1         <= GPR(R,ot3i)
     alu.opr2         <= ot3i(Imm4&Imm8)

     GPR(R,ot3i) <= Aluop.result(alu.opr1,alu.opr2)
     IF (i=0) THEN
        AluStatus <= Aluop.flags(alu.opr1, alu.opr2)
     END IF
END IF
```

Table 7-4. AluOp Field Definition

AluOp	Function	Pseudocode	Flags Modified			
			Z	C	N	V
0000	add	result = opr1 + opr2	x	x	x	x
0001	add w/carry	result = opr1 + opr2 + C	x	x	x	x
0010	subtract	result = opr1 - opr2	x	x	x	x
0011	subtract w/carry	result = opr1 - opr2 - C	x	x	x	x
0100	xor	result = orp1 XOR opr2	x		x	
0101	and	result = opr1 AND opr2	x		x	
0110	or	result = opr1 OR opr2	x		x	
0111	shift left logical	result = $\text{opr1}_{\text{<-opr2}}$, fill with 0s C = C when opr2 = 0 else C = 0 when opr2 > n else C = opr1(n - opr2); where n = size of opr1	x	x	x	
1000	shift right logical	result = fill with 0, $\text{opr1}_{\text{->opr2}}$ C = C when opr2 = 0 else C = 0 when opr2 > n else C = opr1(opr2 - 1); where n = size of opr1	x	x	x	
1001	shift right arithmetic	result = fill with S, $\text{opr1}_{\text{->opr2}}$ C = C when opr2 = 0 else C = opr1(n - 1) when opr2 > n else C = opr1(opr2 - 1); where n = size of opr1	x	x	x	
1010	rotate right	result = fill with opr1, $\text{opr1}_{\text{->opr2}}$ C = C when opr 2 = 0 else C= opr1(MODn(opr2 - 1)); where n = size of opr1	x	x	x	
1011	compare	opr1 - opr2	x	x	x	x
1100	test	opr1 AND opr2	x		x	
1101	not	result = NOT(opr1)	x		x	
1110	transfer	result = opr2				
1111	reserved	reserved				

Figure 7-8. ot3i Field Definition


GPR Register
hw
0
1
byte
0
1
2
3
ot3i Field
Odd halfword GPR, Immediate field with 0 extend
0
0
Imm12
Even halfword GPR, Immediate field with 0 extend
1
0
Imm12
Odd halfword GPR, Immediate field with sign extend
0
s
Imm12
Even halfword GPR, Immediate field with sign extend
1
s
Imm12
Word GPR, Immediate field with 0 extend
2
0
Imm12
Word GPR, Immediate field with sign extend
3
s
Imm12

7.3.4.2 Logical Immediate Opcode

The logical immediate opcode performs the logical functions AND, OR, XOR, and TEST on a GPR register in which the second operand is a 16-bit immediate value. the R field specifies the word GPR operand and destination GPR, and the h field specifies the even (h = 0) or odd (h = 1) halfword of this GPR. The immediate operand is represented by Imm16. The arithmetic function performed is given in the Lop field. Logical opcodes can be performed without changing the ALU status flags if the i field is a 1.

pseudocode:

```
IF (Cond) THEN
     alu.opr1          <= GPR(R:h)
     alu.opr2          <= Imm16

     GPR(R:h) <= Lop.result(alu.opr1,alu.opr2)
     IF (i=0) THEN
        AluStatus <= Lop.flags(alu.opr1, alu.opr2)
     END IF
END IF
```

Table 7-5. Lop Field Definition

LOp	Function	Pseudocode	Flags Modified			
			Z	C	N	V
00	xor	result = orp1 XOR opr2	x		x	
01	and	result = opr1 AND opr2	x		x	
10	or	result = opr1 OR opr2	x		x	
11	test	opr1 AND opr2	x		x	

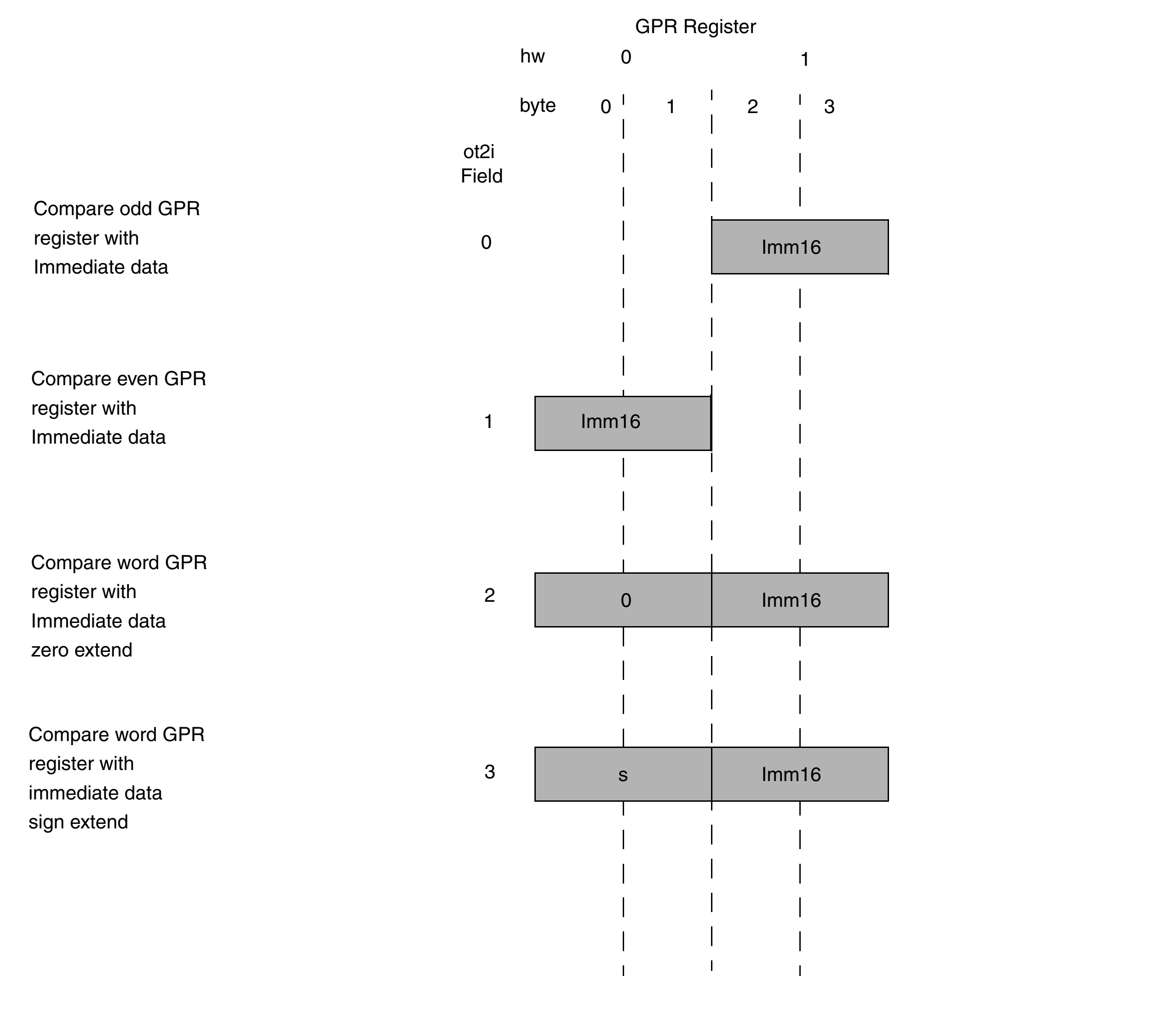
7.3.4.3 Compare Immediate Opcode

The compare immediate opcode performs the compare functions on a GPR register in which the second operand is a 16-bit immediate value. The source GPR operand and destination GPR are specified in the R field, and the immediate operand is represented by Imm16. The actual portion of the GPR and extension of the immediate operand are shown in *Figure 7-9*. The compare immediate opcode always changes the ALU status flags.

pseudocode:

```
IF (Cond) THEN
     alu.opr1          <= GPR(R,ot2i)
     alu.opr2          <= ot2i(Imm16)

     AluStatus <= compare(alu.opr1, alu.opr2)
END IF
```

Figure 7-9. ot2i Field Definition: Compare Halfword/Word Immediate



7.3.4.4 Load Immediate Opcode

The operation arguments of the load immediate opcode are a combination of a GPR register and a 16-bit immediate field. The source GPR operand and destination GPR are specified by the R field and the immediate operand is represented by Imm16. The actual portion of the GPR and extension of the immediate operand are shown in *Figure 7-10* and *Figure 7-11*. Load immediate opcode never changes the ALU status flags if executed.

pseudocode:

```
IF (Cond) THEN
     GPR(R,ot4i) <= ot4i(Imm16)
END IF
```

Figure 7-10. ot4i Field Definition Load Immediate Halfword/Word

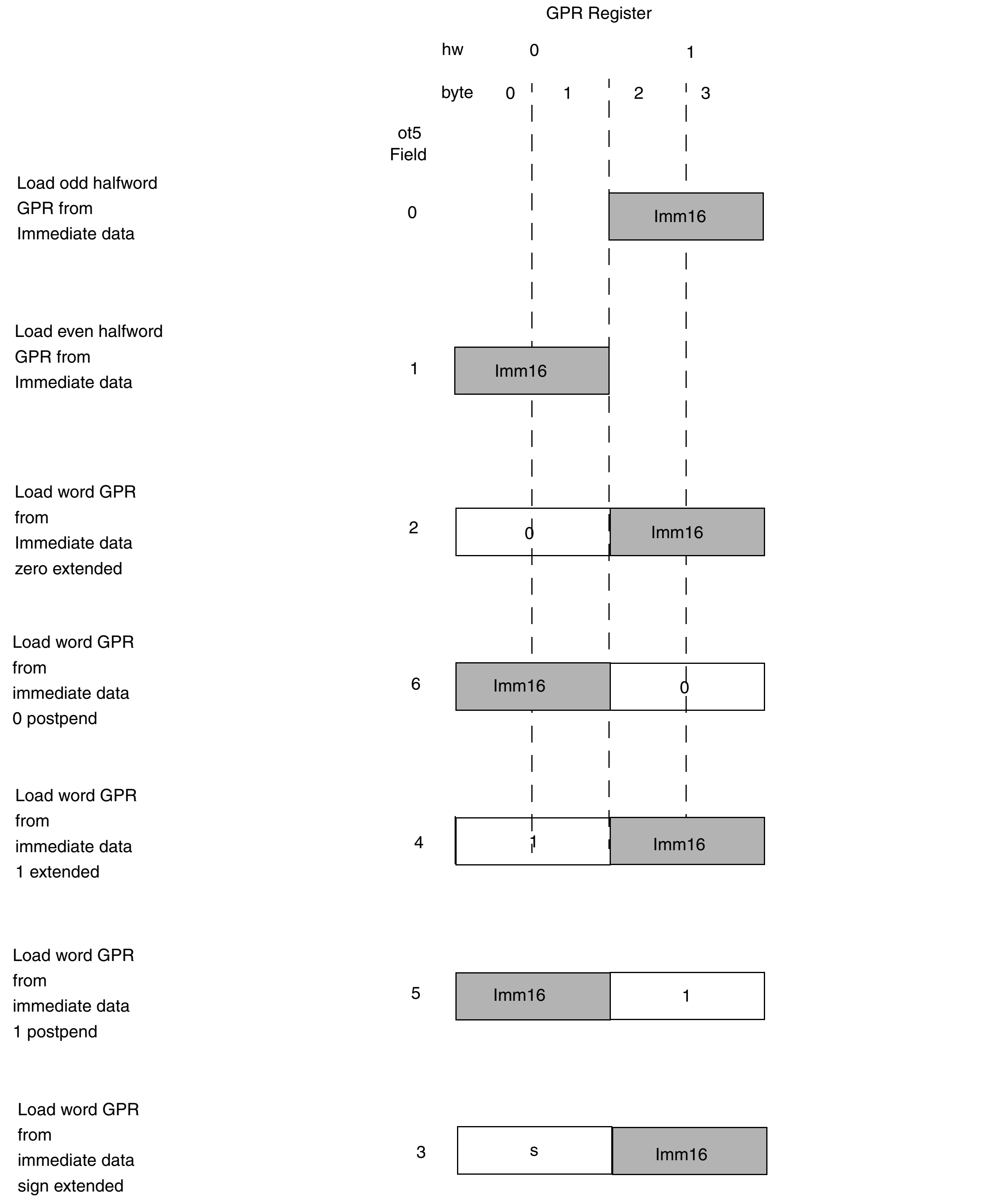
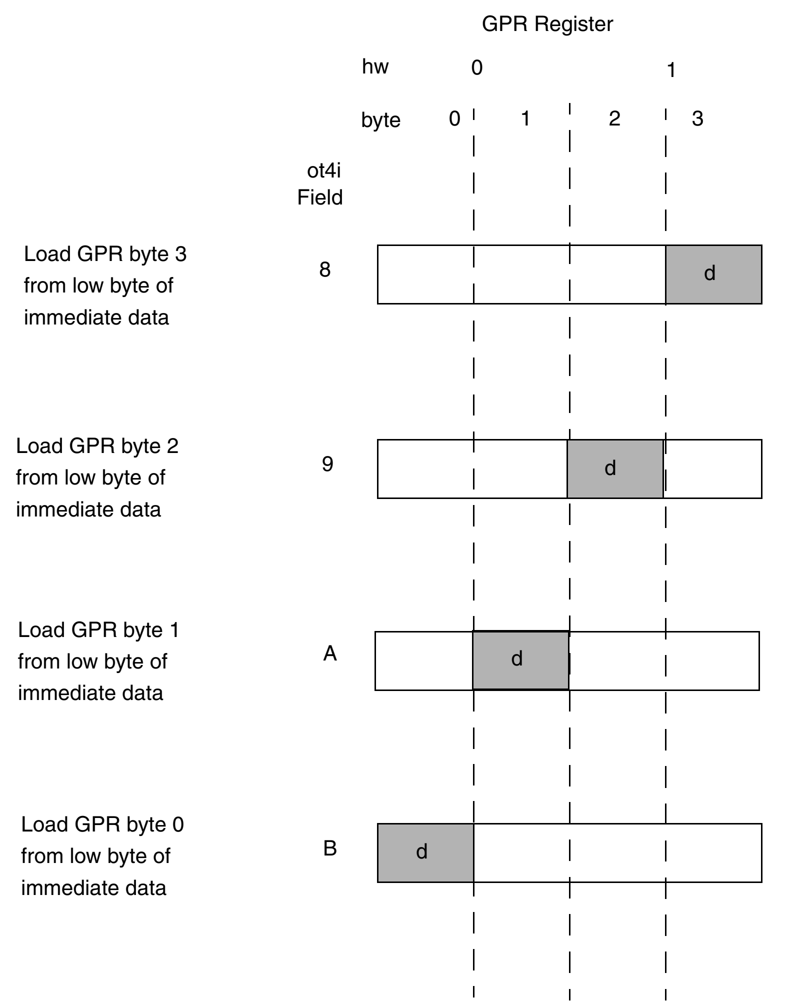


Figure 7-11. ot4i Field Definition: Load Immediate Byte


GPR Register
hw
0
1
byte
0
1
2
3
ot4i Field
Load GPR byte 3 from low byte of immediate data
8
d
Load GPR byte 2 from low byte of immediate data
9
d
Load GPR byte 1 from low byte of immediate data
A
d
Load GPR byte 0 from low byte of immediate data
B
d

7.3.4.5 Arithmetic Register Opcode

The arithmetic register opcode performs an arithmetic function on a GPR register in which the second operand is also a GPR register. The first GPR operand and the destination GPR are specified by the word address R1 and the specific portion to be used is encoded in ot3r. The second operand source is represented by word address R2 with the h field determining the even or odd halfword if the ot3r field indicates a halfword is needed. If the AluOp is a shift or rotate command, the second operand source is used as the amount of the shift or rotate. Otherwise, ot3r indicates the relationship between the two operands. If the AluOp is a logical operation (AND, OR, XOR, TEST), the second operand source can then further be modified by the m and n fields. The m field is the mask field and, if active, creates a 1-bit mask in which the “1” bit is represented by the second operand source. The n field is the invert field and, if active, will invert the second operand source. If both the m and n fields are “1”, the mask is created before the inversion. *Table 7-6* show how to use these fields to create other operations from the basic logic commands. The arithmetic function performed is given in the AluOp field. Arithmetic opcodes can be performed without changing the ALU status flags if the “i” field is set to ‘1’. Arithmetic opcodes can be performed without writing back the data if the “w” field is set to '1'.

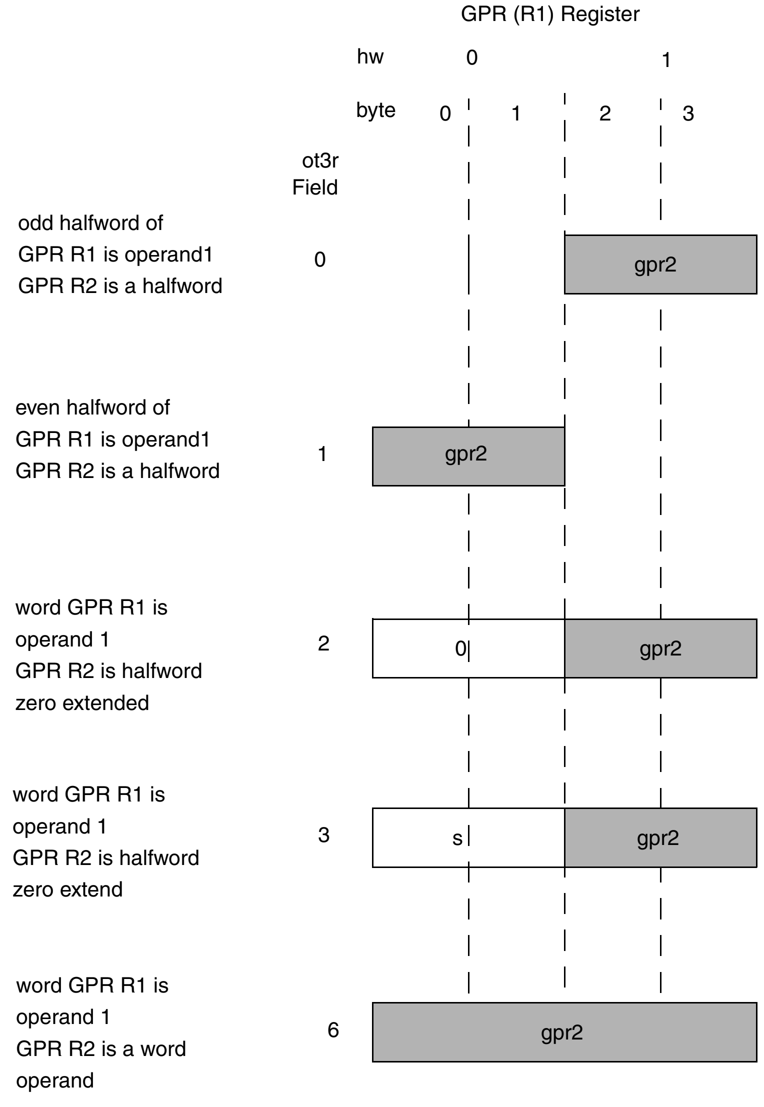
pseudocode:

```
IF (Cond) THEN
     alu.opr1         <= GPR(R1,ot3r)
     alu.opr2         <= GPR(R2,h)

     IF ‘m’=’1’ THEN
        alu.opr2 <= bitmask(alu.opr2)
     END IF

     IF ‘n’=’1’ THEN
       alu.opr2 <= NOT(alu.opr2)
     END IF

     IF 'w' = '0' THEN
        GPR(R2,ot3i) <= Aluop.result(alu.opr1,alu.opr2)
     END IF;

     IF (i=0) THEN
        AluStatus <= Aluop.flags(alu.opr1, alu.opr2)
     END IF
END IF
```

Table 7-6. Arithmetic Opcode Functions

Function	AluOp	m	n
Bit clear	AND	1	1
Bit set	OR	1	0
Bit flip	XOR	1	0

Figure 7-12. ot3r Field Definition


GPR (R1) Register
hw
0
1
byte
0
1
2
3
ot3r Field
odd halfword of GPR R1 is operand1 GPR R2 is a halfword
0
gpr2
even halfword of GPR R1 is operand1 GPR R2 is a halfword
1
gpr2
word GPR R1 is operand 1 GPR R2 is halfword zero extended
2
0
gpr2
word GPR R1 is operand 1 GPR R2 is halfword zero extend
3
s
gpr2
word GPR R1 is operand 1 GPR R2 is a word operand
6
gpr2

7.3.4.6 Count Leading Zeros Opcode

The count leading zeros opcode returns the number of zeros from left to right until the first 1-bit is encountered. This operation can be performed on a halfword or word GPR register. There is a second variation of this command in which the return value of this command is the bit position of the first “1” from left to right in the GPR. The GPR to be analyzed is determined by the R fields. If the GPR is a halfword instruction, the h field indicates whether the even or odd half is used. The destination register is always a halfword register and is represented by the Rd field and halfword indicator ‘hd’. The w field indicates whether the operation is a word or halfword. The n field determines if the command counts the number of leading zeros (n = 0) or if the command returns the bits position of the first one (n = 1). The only flag for this command is the overflow flag, which is set if the GPR being tested contains all zeros. The setting of this flag can be inhibited if the i field is a one.

pseudocode:

```
IF (Cond) THEN
    alu.opr1         <= GPR(Rs,h)

    alu.result <= count_zero_left_to_right(alu.opr1)

    IF ‘n’=’1’ THEN
      GPR(Rd,hd) <= NOT(alu.result)
    ELSE
      GPR(Rd,hd) <= alu.result
    END IF

    IF (i=0) THEN
       AluStatus <= Aluop.flags(alu.opr1, alu.opr2)
    END IF
END IF
```

7.4 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 require a large amount of serialized picocode. DPPU 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:

Tree Search Engine	See *Section 8. Tree Search Engine on page 285*.
Data Store	See *Section 7.4.2* beginning on page 203.
Control Access Bus (CAB) Interface	See *Section 7.4.3* beginning on page 220.
Enqueue	See *Section 7.4.4* beginning on page 223.
Checksum	See *Section 7.4.5* beginning on page 241.
String Copy	See *Section 7.4.6* beginning on page 246.
Policy	See *Section 7.4.7* beginning on page 247.
Counter	See *Section 7.4.8* beginning on page 248.
Coprocessor Response Bus	See *Section 7.4.9* beginning on page 252.
Semaphore	See *Section 7.4.10* beginning on page 253.

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 7-7*. 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 7-7. 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
10	Coprocessor Response Bus (CRB)
11	Semaphore

7.4.1 Tree Search Engine Coprocessor

The Tree Search Engine (TSE) coprocessor 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 tree (SMT).

For complete information, see *Section 8. Tree Search Engine on page 285*.

7.4.2 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 IEW. The Data Store coprocessor also receives configuration information during the dispatch of a timer event or interrupt.

Each thread supported by the Data Store coprocessor has one set of scalar registers and arrays defined for it. The scalar registers control the accesses between the shared memory pool and the ingress and egress data stores. The Data Store coprocessor maintains them. The arrays are defined in the shared memory pool (see *7.2.4 Shared Memory Pool* on page 172):

- The DataPool, which can hold eight quadwords
- Two scratch memory arrays, which hold eight and four quadwords respectively
- The configuration quadword array, which holds port configuration data dispatched to the thread.

The shared memory pool arrays function 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 (one to four 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. These commands are detailed in *Section 7.4.2.2 Data Store Coprocessor Commands* on page 210.

7.4.2.1 Data Store Coprocessor Address Map

Table 7-8. Data Store Coprocessor Address Map (A thread's scalar registers and arrays that are mapped within the Data Store coprocessor) (Page 1 of 2)

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 *7.2.4 Shared Memory Pool* on page 172.)
CCTA	x'02'	19	R/W	Current address of the ingress data buffer 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	Indicates the QW address used for both the Datapool and the QW in the datastore buffer/twin. See *EDIRTY (Update Egress Dirty Quadwords) Command on page 218*, *IDIRTY (Update Ingress Dirty Quadwords) Command on page 219*, *RDMOREE (Read More Quadword From Egress) Command on page 215*, and *RDMOREI (Read More Quadword From Ingress) Command on page 217* for details.
iProtocolType	x'04'	16	R	Layer 3 Protocol identifier set by the hardware classifier (see *7.7 Hardware Classifier* on page 264).
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.
BCI2Byte	x'06'	14/20	R/W	A special-purpose register. The picocode running in the CLP writes a BCI value, but when the register is read, it returns the 14-bit byte count represented by the BCI. The format of the BCI can be changed by configuration (see *Section 13.3 BCI Format Register* on page 444). The BCI value input to this register must match the configured format.
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

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

Table 7-8. Data Store Coprocessor Address Map (A thread's scalar registers and arrays that are mapped within the Data Store coprocessor) (Page 2 of 2)

Name	Register (Array)[1] Number	Size (bits)	Access	Description
BCI2Byte_WChk	x'0E'	24	W	The BCI2Byte_WChk write only register is added. BCI values (the format of which can be configured (see *Section 13.3 BCI Format Register* on page 444) written to this register are checked and a byte count value is generated. The results are read from the BC2Byte register. When an invalid BCI value is written, a result of zero is returned. An invalid BCI is defined in one of the following ways: • Number of data buffers (weight) = 0 • Number of data buffers (weight) = 1 and ending byte location < starting byte location • Starting byte location or ending byte location < 6
ConfigQW	x'FC' (0)	128	R/W	Port Configuration Table Entry for this frame
ScratchMem0	x'FD' (1)	512	R/W	User Defined Array (use to store temporary information, build new frames, and so on)
ScratchMem1	x'FE' (2)	1024	R/W	User Defined Array (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.

The DataPool for Ingress Frames

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 7-9*.

Table 7-9. 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 buffer control block (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).

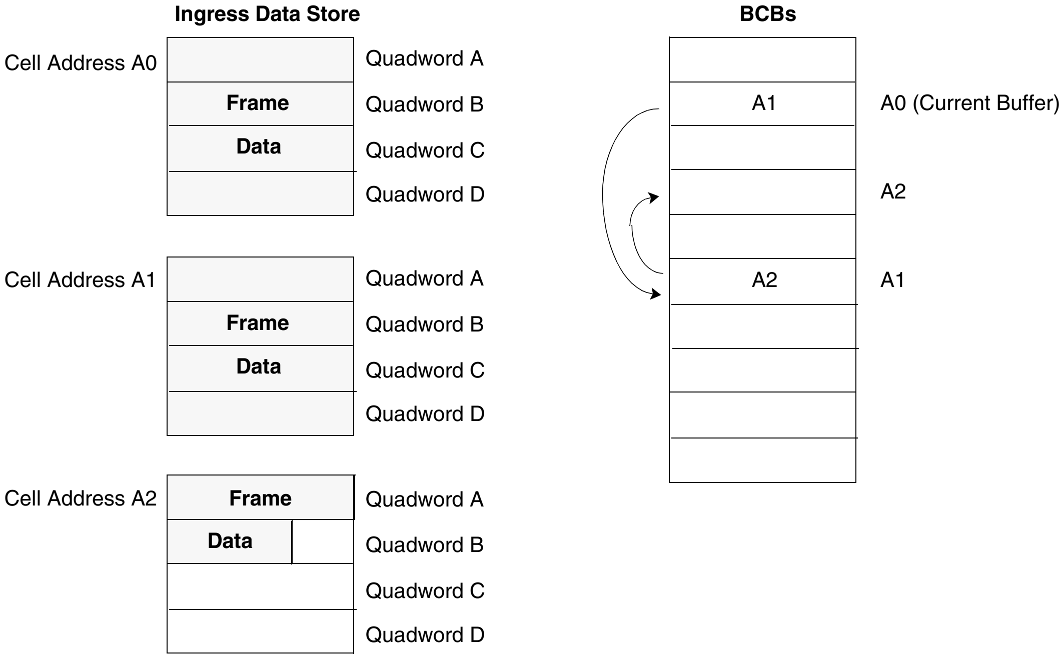
The Ingress Data Store can also be written and read with an absolute address. The address is a BCB address.

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, and the number of quadwords to be transferred.

Figure 7-13 shows an example of a frame stored in the Ingress Data Store:

Figure 7-13. A Frame in the Ingress Data Store



The DataPool for Egress Frames

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 IEW cell and stored in the Egress Data Store (see *Figure 7-14*). General-purpose register (GPR) R0 is initialized during dispatch as shown in *Table 7-10: Egress Frames DataPool Quadword Addresses* on page 208. 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 7-14. Frame in the Egress Data Store (Illustrating the effects of different starting locations)

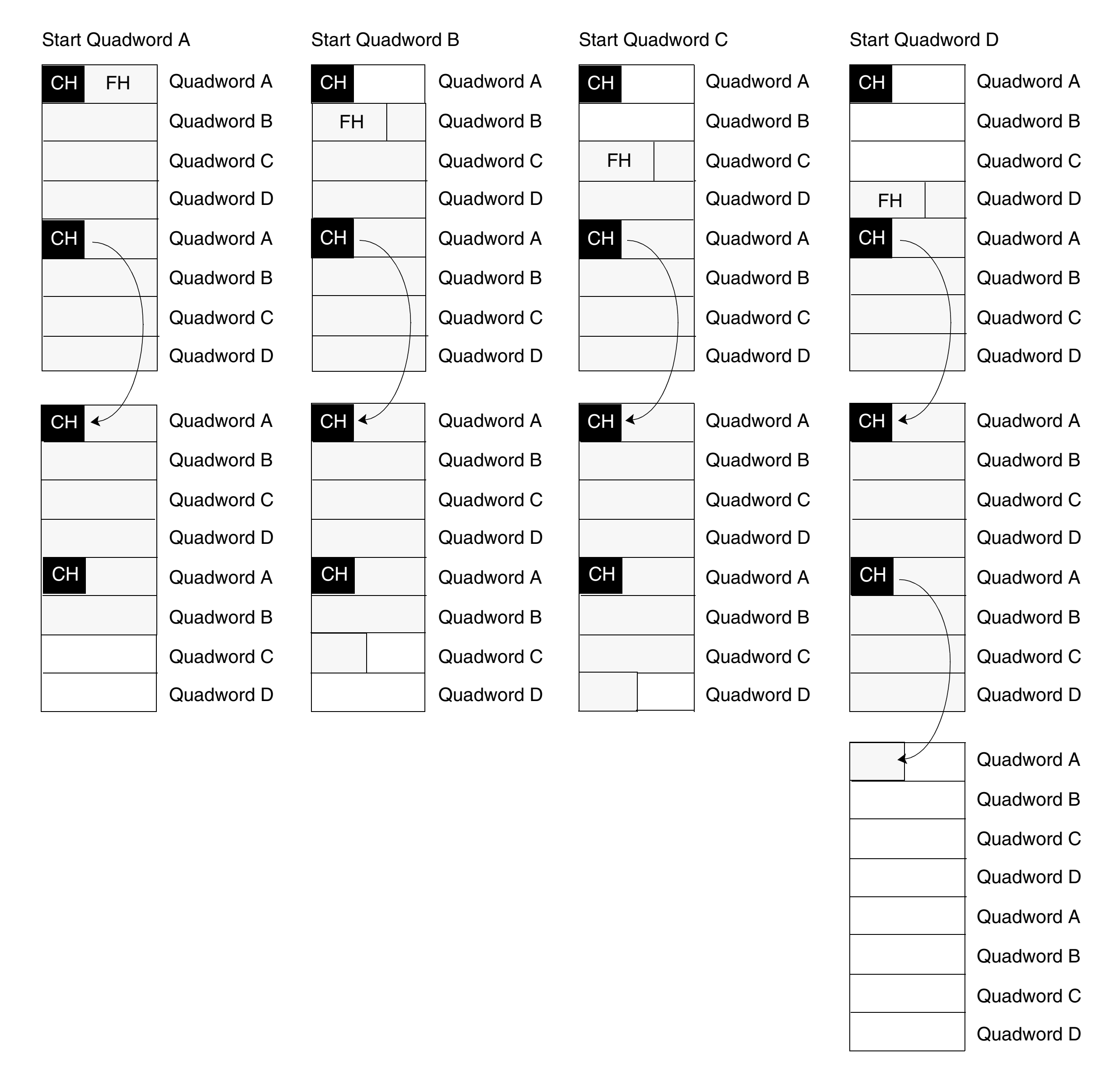


Table 7-10. Egress Frames DataPool Quadword Addresses

Frame Start In Twin Buffer	Frame Quadword Address in the DataPool	GPR R0
Quadword A	0	0
Quadword B	1	10
Quadword C	2	26
Quadword D	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 side, the Egress DataPool contains cell headers. When the exact content of the twin-buffer is copied into the DataPool, it may include the 6-byte NP2G 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 7-11*.

Table 7-11. 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 7-12* shows the number of frame-bytes in the DataPool after a frame dispatch.

Table 7-12. 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 7-12* 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.

As a general rule, each set of four quadwords provides exactly 58 bytes.

7.4.2.2 Data Store Coprocessor Commands

The Data Store coprocessor provides the following commands:

Command	Opcode	Description
WREDS	0	Write Egress Data Store. Enables 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). For more information, see *WREDS (Write Egress Data Store) Command* on page 212.
RDEDS	1	Read Egress Data Store. Enables 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). For more information, see *RDEDS (Read Egress Data Store) Command* on page 213.
WRIDS	2	Write Ingress Data Store. Enables the CLP to write data to the Ingress data store (in multiples of quadwords only). For more information, see *WRIDS (Write Ingress Data Store) Command* on page 214.
RDIDS	3	Read Ingress Data Store. Enables the CLP to read data from the Ingress Data Store (in multiples of quadwords only). For more information, see *RDIDS (Read Ingress Data Store) Command* on page 214.

Command	Opcode	Description
RDMOREE	5	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 DataPool is essentially a map of a twin's content, the frame data might wrap within the DataPool; the picocode keeps track of the data’s location within the DataPool. For more information, see *EDIRTY (Update Egress Dirty Quadwords) Command* on page 218
RDMOREI	7	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. For more information, see *RDMOREI (Read More Quadword From Ingress) Command* on page 217.
LEASETWIN	8	Lease Twin Buffer. Returns the address of a free twin buffer. Use this command when creating new data in the Egress Data Store. For more information, see*7.4.3 Control Access Bus (CAB) Coprocessor on page 220*
EDIRTY	10	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 enables 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. For more information, see *EDIRTY (Update Egress Dirty Quadwords) Command* on page 218
IDIRTY	12	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 enables 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. For more information, see *IDIRTY (Update Ingress Dirty Quadwords) Command* on page 219

Note: The Egress Data Store has a longer access time than the Ingress Data Store. For better performance, as much frame parsing should be done on the Ingress side 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, based on twin format when toggle mode is enabled, or if the decision is made automatically by information in the Dispatch DSU register.

Table 7-13. WREDS Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 when toggle mode is disabled: 00 Writes to the default DSU 01 Writes to DS0 10 Writes to DS1 11 Writes to DS0 and DS1 When toggle mode is enabled: 00 Uses the twin format determined at dispatch and captured in the Disp_DSU 01 Forces twin format 0 10 Forces twin format 1 11 Invalid
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 QW 0-3 of the first buffer of the twin. 100 - 111 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 *7.2.4 Shared Memory Pool* on page 172).

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

Table 7-14. WREDS Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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, based on twin format when toggle mode is enabled, or if this decision is made automatically by information in the Dispatch DSU register.

Table 7-15. RDEDS Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 when toggle mode is disabled: 00 Reads from the default DS 01 Reads from DS0 10 Reads from DS1 11 Reserved When toggle mode is enabled: 00 Uses the twin format determined at dispatch and captured in the Disp_DSUSel 01 Forces twin format 0 10 Forces twin format 1 11 Invalid
Disp_DSUSel	1	R		Indicates the default DS chosen at dispatch. When toggle mode is disabled: 0 DS0 1 DS1 When toggle mode is enabled (see *Section 13.2 Toggle Mode Register* on page 444), indicates the twin format: 0 Twin format 0 1 Twin format 1
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 *7.2.4 Shared Memory Pool* on page 172).

When DSControl is set to 00, the quadwords are read from the default DS or the default twin format when toggle mode is enabled, which is determined based on the Dispatch DSUSel field.

Table 7-16. RDEDS Output

Name	Size	Operand Source		Description
		Direct	Indirect	
OK/KO	1	Flag		KO - An error occurred when reading the fourth quadword of a twin buffer indicating the link pointer contained in the data had a parity error. OK - Indicates that the link pointer had valid parity or that the link pointer wasn't read on this access.
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 7-17. WRIDS Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 *7.2.4 Shared Memory Pool* on page 172).

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 7-18. WRIDS Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-19. RDIDS Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 *7.2.4 Shared Memory Pool* on page 172).

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 7-20. RDIDS Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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.

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 link pointer 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 7-21. RDMOREE Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 when toggle mode is disabled: 00 Reads from the default DS 01 Reads from DS0 10 Reads from DS1 11 Reserved When toggle mode is enabled: 00 Uses the twin format determined at dispatch and captured in the Disp_DSU 01 Forces twin format 0 10 Forces twin format 1 11 Invalid
Disp_DSUSel	1	R		Indicates the default DS chosen at dispatch when toggle mode is disabled: 0 DS0 1 DS1 When toggle mode is disabled (see *Section 13.2 Toggle Mode Register* on page 444), indicates the twin format: 0 Twin format 0 1 Twin format 1
CCTA	19	R		Current Cell/Twin Address. Contains the twin source address in the Egress Data Store. This register is initialized at dispatch to point to the current twin fetched at dispatch.
NQWA	3	R		Next Quadword Address. The contents of this register indicates both the target quadword address in the Datapool as well as the source quadword of the twin buffer. This register is initialized at dispatch to point to the next QW after the data fetched at dispatch.

Table 7-22. RDMOREE Output

Name	Size	Operand Source		Description
		Direct	Indirect	
OK/KO	1	Flag		KO - An error occurred when reading the fourth quadword of a Twin buffer indicating that the Link Pointer contained in the data had a parity error. OK - Indicates that the link pointer had valid parity or that the link pointer wasn't read on this access.
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.

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 7-23. RDMOREI Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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. Contains the source ingress data buffer address in the Ingress Data Store. This register is initialized at dispatch to point to the next data buffer after the data fetched at dispatch.
NQWA	3	R		Next Quadword Address. The contents of this register indicates both the target quadword address in the Datapool as well as the source quadword of the ingress data buffer. The low two bits indicate the QW in the ingress data buffer, while all three bits are used to indicate the QW in the datapool. This register is initialized at dispatch to point to the next QW after the data fetched at dispatch.

Table 7-24. RDMOREI Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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.

LEASETWIN Command

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

Table 7-25. LEASETWIN Output

Name	Size	Operand Source		Description
		Direct	Indirect	
OK/KO	1	Flag		KO - No twins are currently available. OK - The command was successful.
CCTA	19	R		Current Cell/Twin Address. Contains the address of the newly leased twin.

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.

Table 7-26. EDIRTY Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
CCTA	19	R		Current Cell/Twin Address. Contains the twin target address in the Egress Data Store.
NQWA	3	R		Next Quadword Address. The contents of this register indicates which QWs are considered for EDirty processing. NQWA Datapool QW Target Twin QW 0 7-0 7-0 1 0 0 2 1-0 1-0 3 2-0 2-0 4 3-0 3-0 5 4-0 4-0 6 5-0 5-0 7 6-0 6-0

Table 7-26. EDIRTY Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
DirtyQW	8	R		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 when toggle mode is disabled: 00 Writes to the default DSU 01 Writes to DS0 10 Writes to DS1 11 Writes to DS0 and DS1 When toggle mode is enabled: 00 Uses the twin format determined at dispatch and captured in the Disp_DSU 01 Forces twin format 0 10 Forces twin format 1 11 Invalid
Disp_DSU	2	R		The default DSU, initialized at dispatch, when toggle mode is disabled: 00 Invalid 01 DS0 10 DS1 11 Both DS0 and DS1 When toggle mode is enabled: 00 Invalid 01 Twin format 0 10 Twin format 1 11 Invalid

Table 7-27. EDIRTY Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-28. IDIRTY Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
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 contents of this register indicates which QW are considered for action by the IDirty command. NQWA / Datapool QW / Target Ingress data buffer QW 0 / 7-4 / 3-0 1 / 0 / 0 2 / 1-0 / 1-0 3 / 2-0 / 2-0 4 / 3-0 / 3-0 5 / 4 / 0 6 / 5-4 / 1-0 7 / 6-4 / 2-0
DirtyQW	8	R		Indicates which quadwords need to be updated

Table 7-29. IDIRTY Output

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

7.4.3 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.

7.4.3.1 CAB Coprocessor Address Map

The CAB coprocessor has three scalar registers that are accessible to a thread and are shown in *Table 7-30*:

Table 7-30. 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

7.4.3.2 CAB Access to NP2G Structures

The Control Address Bus (CAB) is the NP2G'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 7-31. 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 7-32. 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’	Reserved	
‘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.

7.4.3.3 CAB Coprocessor Commands

The CAB Coprocessor provides the following commands:

Command	Opcode	Description
CABARB	0	CAB Arbitration. 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. For more information, see *CABARB (CAB Arbitration) Command* on page 222
CABACCESS	1	Access CAB. Moves data onto or from the CAB and the attached CAB accessible registers. The source and destination within the DPPU are GPRs. For more information, see *CABACCESS Command* on page 223
CABPREEMPT	3	Preempt CAB. Used only by the GFH thread, it enables 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. For more information, see *CABPREEMPT Command* on page 223

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 7-33. CABARB Input

Name	Size	Operand Source		Description
		Direct	Indirect	
Start_NEnd	1	Imm16(0)		1 Start arbitration 0 Release 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 7-34. CABACCESS Input

Name	Size	Operand Source		Description
		Direct	Indirect	
Read_NWrite	1		Imm12(0)	1 Perform a CAB read 0 Perform a CAB write
Address	32		GPR(31..0)	The CAB address
CABData	32	R		Load for CAB write command

Table 7-35. CABACCESS Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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.

7.4.4 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.

7.4.4.1 Enqueue Coprocessor Address Map

Table 7-36. 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 contains dispatch and enqueue information (fields) and is stored in three FCBPage arrays (ActiveFCBPage1, InActiveFCBPage2, and FCBPage2). There are two FCBPage formats, ingress and egress. The FCBPage format defines fields on byte boundaries. Information that is not an integral number of bytes is right-justified in a field sized to accommodate the information. Since the FCBPage is stored in arrays within the Shared Memory Pool, fields within the FCBPage can be accessed using all of the standard methods of accessing arrays.

Figure 7-15. 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
Reserved (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 7-37. Ingress FCBPage Description (Page 1 of 3)

Field	FCB Page Offset	Initialized by Dispatch Unit	Enqueue Info	Size (bits)	Description
SP	x‘00’	Y	N	6	Source Port is the port identifier where this frame was received.
Abort	x‘01’	Y	N	1	Aborted Frame indicates that the frame had been marked abort at the time of Dispatch.
GT	x‘02’	Y	N	1	Guided Traffic indicator. This bit must be set to ‘1’ when the frame is guided traffic.
FCInfo	x‘03’	Y	Y	4	Flow Control color and frame drop information. See *Table 7-89: Flow Control Information Values* on page 269. Setting this field to x‘F’ disables flow control. Flow control must be disabled when enqueing to the GDQ, GFQ, or the discard queue.
WBC	x‘04’	Y	N	15	Working byte count. This is 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. Encode is TDMU(1:0) DMU 00 A 01 Reserved 10 C 11 D

Table 7-37. Ingress FCBPage Description (Page 2 of 3)

Field	FCB Page Offset	Initialized by Dispatch Unit	Enqueue Info	Size (bits)	Description
L3Stk/iDSU	x‘0D’	N	Y	8/4	L3Stk (8-bit field). 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 (4-bit field). Data Store unit field in the frame header. Indicates where the frame should be stored when entering the egress side when toggle mode is disabled. When iUCnMC = 1 (frame is unicast). When set to 0, hardware determines the value of the Data Store unit field by using the value of the TDMU field and contents of the Ingress TDMU Data Storage Map Register (I_TDMU_DSU) (see *13.13 Ingress Target DMU Data Storage Map Register (I_TDMU_DSU)* on page 459).
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 IEW.
TOS	x‘0F’	Y	Y	8	If the frame is an IP frame these bits will be the TOS field in the IP Header, otherwise they will be initialized to 0’s. For differentiated services, the following subfields are defined: 7:5 AF Class. Bits are used by the flow control hardware when addressing the Transmit Probability table. 4:2 Drop precedence 1:0 CU (currently unused).
Reserved	x‘10’	N	Y	16	Reserved.
iUCnMC	x‘12’	N	Y	1	Unicast/Multicast indicator 0 Multicast frame 1 Unicast frame
Priority_SF	x‘13’	N	Y	2	Priority and special field indicators defined as: Bit Description 0 Special Field. When set to ‘1’, indicates enqueues to the discard queue, GDQ, or I-GFQ. This is normally set by hardware when the QCLASS is used in the ENQI command. Flow control is disabled when this field is set to ‘1’. 1 Priority. Ingress scheduler priority indicates the user priority assigned to the frame. High priority is indicated by a value of 0.
LID/MID	x‘14’	N	Y	21/17	Lookup ID (21-bit field). Ingress picocode uses to pass information to the egress picocode. Used when iUCnMC = 1 (frame is unicast). Multicast ID (17-bit field). Ingress picocode uses to pass information to the egress picocode. Used when iUCnMC = 0 (frame is multicast).
VLANHdr	x‘18’	N	Y	16	VLAN Header. This is the Tag Control Information field defined in the IEEE 802.3 standard.
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

Table 7-37. Ingress FCBPage Description (Page 3 of 3)

Field	FCB Page Offset	Initialized by Dispatch Unit	Enqueue Info	Size (bits)	Description
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. A 4-byte field whose contents are defined by picocode.
CounterControl	x‘28’	N	Y	14	Passed to Flow Control for delayed counter manager functions. Bits Description 13 When set to 1 enables counter operation for this enqueue. Counter updates on an enqueue work only when the target is the egress side. Counter updates do not occur when enqueueing to the GDQ, GFQ, or the discard queue. 12 Add/$\overline{\text{Increment}}$ 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 7-16. Egress FCBPage Format

0	1	2	3	4	5	6	7
MPLS_3 (8)	eUCMC (3)	DSUSel(1) MPLS_4 (8)	FCInfo (4)	BCI (20/24)			
8	9	10	11	12	13	14	15
CurrTwin (20)				Type (3) MPLS_5 (8)	DSU (2)	QHD (1)	OW (7)
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)			
DAPtr (6)	SAPTr (6)	MPLS_6 (8)	MPLS_7 (6)	MPLS_8 (8)	MPLS_9 (8)	MPLS_10 (8)	MPLS_11 (8)
32	33	34	35	36	37	38	39
EN_HWA (6)	VLAN_MPLS_HWA (5)	CRCAction (2)	DLLStake (6)	TTLAssist (2)	MPLS_0 (8)	MPLS_1 (8)	MPLS_2 (8)
40	41	42	43	44	45	46	47
CounterControl (14)		CounterData (16)		CounterBlockIndex (20)			

Table 7-38. Egress FCBPage Description (Page 1 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
MPLS_3	x‘00’	N	8	MPLS insert byte 3 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
eUCMC	x‘01’	Y	3	Egress Unicast/Multicast bits 000 Unicast The frame is enqueued a single time. The twins used to hold the data for this frame are released to the free pool after transmission. 001 First Multicast The first enqueue for a frame that is enqueued multiple times. The frame data is not replicated; all instances of the frame transmission use the same source in the Data Store, but frame alterations might be different. The last instance of the multicast frame to be transmitted releases the twins back to the free pool. 010 Middle Multicast The middle enqueues for a frame that is enqueued multiple times. There may not be a middle frame if the frame is not transmitted more than twice. 011 Last Multicast The last enqueue for a frame that is enqueued multiple times. 100 Unicast Static Frame enqueueThe frame is enqueued a single time. The twins used to hold the data are not released to the free pool after transmission. 101 First Multicast Static FrameSimilar to '001', but the twins used to hold the data are not released to the free pool. 110 Middle Multicast Static FrameSimilar to '010', but the twins used to hold the data are not released to the free pool. 111 Last Multicast Static FrameSimilar to '011', but the twins used to hold the data are not released to the free pool.
DSUSel	x‘02’	Y	1	This field is initialized to indicate which DSU was used on dispatch but is not read back by hardware. To override the DSU, the picocode must use the Disp_DSU or Disp_DSUSel registers. (See *Table 7-8* on page 204.) When toggle mode is disabled: 0 DS0 1 DS1 When toggle mode is enabled, frame data is placed into DS0 and DS1 based on twin format: 0 Twin format 0 1 Twin format 1
MPLS_4	x‘02’	N	8	MPLS insert byte 4 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
FCInfo	x‘03’	Y	4	Flow control color information pulled from the frame header by the hardware classifier. See *Table 7-89: Flow Control Information Values* on page 269.

Table 7-38. Egress FCBPage Description (Page 2 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
BCI	x‘04’	Y	24	Byte count indicator passed from a queue (GR0, GR1, GB0, GB1, GPQ, GTQ, GFQ) to the FCBPage. Indicates starting byte location within the first twin, number of data buffers, and ending byte location within the last twin. The format of this field is controlled by the BCI Format control (see *Section 13.3 BCI Format Register* on page 444). In earlier versions, the format of the BCI is always the compact format described below. BCI_Format_Ctl Description 0 Compact format. Bit Description 19:14 Starting Byte (of user data, not frame header) 13:6 Number of Data Buffers (Weight) 5:0 Ending Byte 1 Expanded format. Bit Description 21:16 Starting Byte (of user data, not frame header) 15:8 Number of Data Buffers (Weight) 7:6 Reserved 5:0 Ending Byte Byte numbering within the data buffers starts at 0 and goes to 63. An invalid BCI specification might result in unpredictable behavior. An invalid BCI is defined in one of the following ways: • Number of data buffers (weight) = 0 • Number of data buffers (weight) = 1 and ending byte < starting byte • Starting byte or ending byte < 6
CurrTwin	x‘08’	Y	20	At dispatch, indicates the first twin address of the frame
Type	x‘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 when toggle mode is disabled: 0 DSU0 1 DSU1 Type(0) for GR0, GB0 when toggle mode is disabled: 0 DSU0 1 Both DSU0 and 1 Type(0) GR1, GB1 when toggle mode is disabled: 0 DSU1 1 Both DSU0 and 1 Type(0) for GR0, GR1, GB0, GB1, GFQ, GTQ, GPQ when toggle is enabled: 0 Twin format 0 1 Twin format 1
MPLS_5	x‘0C’	N	8	MPLS insert byte 5 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.

Table 7-38. Egress FCBPage Description (Page 3 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
DSU	x'0D'	Y	2	Indicates in which DSU(s) the data for this frame is stored. Value is DSU(1:0). When toggle mode is disabled: Value Description 00 Reserved 01 Stored in DSU 0 10 Stored in DSU 1 11 Stored in both DSU 0 and 1 When toggle mode is enabled (see *Section 13.2 Toggle Mode Register* on page 444), data is placed into DS0 and DS1 based on twin format: 00 Invalid 01 Forces twin format 0 10 Forces twin format 1 11 Invalid
QHD	x'0E'	N	1	Twin Header Qualifier. Indicates type of twin pointed to by CurrTwin. Used for egress frame alteration. 0 The twin pointed to by the CurrTwin is a data twin 1 The twin pointed to by the CurrTwin is a header twin
OW	x'0F'	N	7	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	x'10'	N	20	Queue identifier. The format of the QID is determined by QID(19:18) as follows: QID(19:18) 00 and Scheduler is active, the queue is a Flow QCB QID(10:0) Flow QCBAddress 00 and Scheduler disabled, the queue is a indicates the Target Port where: QID (6) Priority QID (5:0) Target Port ID 11 G queue identifier, where QID(17:0) indicates the queue as: QID (17:0) queue 000 GR0 001 GR1 010 GB0 011 GB1 100 GFQ 101 GTQ 110 GPQ 111 Discard
DATAFirstTwin	'14'	N	19	Address of frame's first data twin. Valid when OW is not 0.

Table 7-38. Egress FCBPage Description (Page 4 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
EtypeAct	x‘15’	N	3	The field is defined as Bits(2:0) Definition 000 Etype value is invalid 001 Etype value is valid and is used when SA/DA insert/overlay hardware assisted frame alteration is active to insert or overlay the Etype field of the frame being processed. 010 EtypeValue is valid and is used when SA/DA insert/overlay hardware assisted frame alteration is active to insert or overlay the Length/Type field of the frame when the EN_HWA(1:0) field is set to either ‘01’ (overlay) or ‘10’ (insert). A 3-byte IP SAP, x‘06 0603’, is inserted or overlaid following the Length/Type field. 011 EtypeValue is valid and is used when SA/DA insert/overlay hardware assisted frame alteration is active to insert or overlay the Length/Type field of the frame when the EN_HWA(1:0) field is set to either ‘01’ (overlay) or ‘10’ (insert). Additionally, an 8-byte IP SNAP, x‘AAAA 0300 0000 0800’, is inserted or overlaid following the Length/Type field. 100 EtypeValue is valid and is used when SA/DA insert/overlay hardware assisted frame alteration is active to insert or overlay the Length/Type field of the frame when the EN_HWA(1:0) field is set to either ‘01’ (overlay) or ‘10’ (insert). Additionally, an 8-byte MPLS UC SNAP header, x'AAAA 0300 0000 8847', is inserted or overlaid following the Length/Type field. 101 EtypeValue is valid and is used when SA/DA Insert/Overlay hardware assisted frame alteration is active to insert or overlay the Length/Type field of the frame when the EN_HWA(1:0) field is set to either ‘01’ (overlay) or ‘10’ (insert). Additionally, 8-byte MPLS MC SNAP header, x'AAAA 0300 0000 8848', is inserted or overlaid following the Length/Type field. 110 - 111 Reserved.
EtypeValue	x‘16’	N	16	Ethertype value used in insert / overlay egress frame alteration
DAPtr	x‘18’	N	6	Destination Address Pointer for egress frame alteration. This field contains the index into the DA_Array. For this field definition to be used to access the DA_Array, bit 2 of EN_HWA must be set to ‘1’.
SAPtr	x‘19’	N	6	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	x‘1A’	N	16	Destination address bits (47:32) used for frame alteration SA/DA insert or overlay actions.
MPLS_6	x‘1A’	N	8	MPLS insert byte 6 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
MPLS_7	x‘1B’	N	8	MPLS insert byte 7 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
DA31_0	x‘1C’	N	32	Destination address bits (31:0) used for frame alteration SA/DA insert or overlay actions.
MPLS_8	x‘1C’	N	8	MPLS insert byte 8 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.

Table 7-38. Egress FCBPage Description (Page 5 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
MPLS_9	x‘1D’	N	8	MPLS insert byte 9 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
MPLS_10	x‘1E’	N	8	MPLS insert byte 10 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
MPLS_11	x‘1F’	N	8	MPLS insert byte 11 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
EN_HWA	x‘20’	N	6	Enable Hardware Assist control. Bits 1:0: 00 No hardware assist. 01 Overlay SA/DA. Overlay Etype as indicated by the EtypeAct field. SA data is found in the SA_Array, as indexed by the SAPtr. DA data is found either in the DA field (DA47_32 and DA31_0) when EN_HWA bit 2 is set to ‘0’, or in the DA_Array, as indexed by the DAPtr, when EN_HWA bit 2 is set to ‘1’. 10 Insert SA/DA. Insert Etype as indicated by the EtypeAct field. SA data is found in the SA_Array, as indexed by the SAPtr. DA data is found either in the DA field (DA47_32 and DA31_0) when EN_HWA bit 2 is set to ‘0’, or in the DA_Array, as indexed by the DAPtr, when EN_HWA bit 2 is set to ‘1’. 11 Reserved. Bit 2 controls use of the DA_Array. When this bit is set to ‘1’, the FCBPage offset x'18' contains the DAPtr and offset x'19' contains the SAPtr (the important data for the SAPtr is always in offset x'19' due to the alignment of the data). Bit 2: 0 FCBPage Offsets x'1A’ to x'1F’ contain immediate DA data and can be used for SA/DA insert or overlay as controlled by EN_HWA. 1 DA data for SA/DA insert, as determined by the value of EN_HWA, is found in the DA_Array and is indexed by the value of DAPtr. FCBPage offsets x'1A’ to x'1F' can be used for MPLS_n byte insert Bit 3 disables hardware assisted padding: Bit 3: 0 Hardware Assisted Pad. When set to ‘0’, enables the DMU to pad the frame to a configured number of bytes with a configured padding byte. See *Section 13.25 Frame Pad Configuration Register (DMU_Pad)* on page 489) for more information. 1 No hardware assist. Bits 5:4 control a two-byte overlay of the protocol field. The protocol field consists of the two bytes preceding the DLLStake byte; the DLLStake must be greater than or equal to two. Bits 5:4: 00 No overlay action. 01 Overlay the protocol field with the PPP-IP protocol value x'0021' 10 Overlay the protocol field with the PPP-MPLS protocol value x'0281' (MPLS unicast) 11 Overlay the protocol field with the protocol value held in the Etype-Value field.

Table 7-38. Egress FCBPage Description (Page 6 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
VLAN_MPLS_HWA	x'21'	N	5	Hardware assist for deleting VLAN Tags, deleting MPLS Labels and performing MPLS Label swap. The field is defined as follows: Bits 1:0: 00 No action 01 MPLS Label Delete. 10 VLAN Tag Delete 11 MPLS Label Swap The location of the VLAN Tag is fixed at offset 12. The location of the MPLS label is determined by the value of the DLL stake field. The MPLS Label Swap function modifies the label stack entry (4 bytes) as follows: - The 20-bit label field is replaced by the contents of the DA(47:28) field - The stack entry Exp and S fields are unchanged. - The stack entry TTL field is decremented. Bits 4:2 define the MPLS_n bytes to be inserted prior to the byte in the received frame indicated by the DLLStake. Bits 4:2: 000 No insert action. 001 MPLS_0 through MPLS_1. 010 MPLS_0 through MPLS_3. 011 MPLS_0 through MPLS_5. 100 MPLS_0 through MPLS_7. DA immediate is not available for use, the DAPtr must be used if insert or overlay SA/DA hardware assist is active. 101 MPLS_0 through MPLS_9. DA immediate is not available for use, the DAPtr must be used if insert or overlay SA/DA hardware assist is active. 110 MPLS_0 through MPLS_11. DA immediate is not available for use, the DAPtr must be used if insert or overlay SA/DA hardware assist is active. 111 MPLS_8 through MPLS_11. DA immediate is not available for use, the DAPtr must be used if insert or overlay SA/DA hardware assist is active.
CRCAction	x'22'	N	2	Egress frame alteration controls for modifying the CRC of a frame. Value Description 00 No operation 01 CRC Delete 10 Append CRC 11 Overlay CRC
DLLStake	x'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	x'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

Table 7-38. Egress FCBPage Description (Page 7 of 7)

Field	FCB Page Offset	Initialized by Dispatch Unit	Size (bits)	Description
MPLS_0	x‘25’	N	8	MPLS insert byte 0 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
MPLS_1	x‘26’	N	8	MPLS insert byte 1 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
MPLS_2	x‘27’	N	8	MPLS insert byte 2 of 11. Enables insertion of an MPLS label of up to 12 bytes. The data is inserted prior to the byte in the received frame indicated by the DLLStake. The number of bytes inserted (starting at MPLS_0) is controlled by the expanded VLAN_MPLS_HWA.
CounterControl	x‘28’	N	14	Passed to Flow Control for delayed counter manager functions Bits Description 13 When set to 1 enables counter operation for this enqueue. Counter updates on an enqueue work only when the target is a target port or flow queue. 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 Counter Definition Table Index
CounterData	x‘2A’	N	16	Data passed to Egress Flow Control for delayed counter manager add functions.
CounterBlockIndex	x‘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 7-37* and *Table 7-38*. Values that are not indicated as initialized at dispatch are initialized to ‘0’ for the “active” FCBPage.

7.4.4.2 Enqueue Coprocessor Commands

The following commands are supported by the Enqueue coprocessor:

Command	Opcode	Description
ENQE	0	Enqueue Egress. Enqueues to the Egress EDS via the Completion Unit. For more information, see *ENQE (Enqueue Egress) Command* on page 236.
ENQI	1	Enqueue Ingress. Enqueues to the Ingress EDS via the Completion Unit. For more information, see *ENQI (Enqueue Ingress) Command* on page 238.
ENQCLR	2	Enqueue Clear. Clears (sets all fields to zero in) the specified FCBPage. For more information, see *ENQCLR (Enqueue Clear) Command* on page 240.
Release_Label	3	Release Label. Releases the label in the Completion Unit for this frame. For more information, see *RELEASE_LABEL Command* on page 240.

ENQE (Enqueue Egress) Command

ENQE enqueues frames to the Egress target queues.

Table 7-39. ENQE Target Queues ENQE command enqueues a frame in one of these Egress target queues

Target Queue	Description	Note
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.	1
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.	1
GB0	Same as GR0, but treated by the Dispatch Unit as lower priority	1
GB1	Same as GR1, but treated by the Dispatch Unit as lower priority	1
GFQ	Enqueued frames in this queue are destined for the GFH	1
GTQ	Enqueued frames in this queue are destined for the GTH	1
GPQ	Enqueued frames in the queue are destined for the embedded PowerPC	1
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.	

1. When enqueuing to the GR0, GR1, GB0, GB1, GFQ, GTQ, or the GPQ it is recommended to check the depth of the queue. A dead lock can occur if an attempt is made to enqueue to a full queue. When the queue is full, the enqueue should be re-directed to the discard queue, an action performed by code, and the event reported either via a count or guided traffic.

The QID field in the FCBPage selects the Egress target queue. *Table 7-40* 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 7-40. 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
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 7-41. 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 7-42*), BCI, MCCA

Table 7-42. 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 7-44*. Like ENQI, ENQE does not modify the FCBPage.

Table 7-43. ENQE Command Input

Name	Size	Operand Source		Description
		Direct	Indirect	
QueueClass	5	Imm16(4..0)	GPR(4..0)	Egress Queue Class as defined in *Table 7-44*.
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 7-44. 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 7-45. 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 TDMU 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. That is, frames are discarded at 58 bytes per slot.
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 7-46* shows the coding of this selection.

Table 7-46. 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	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 7-47* shows the parameters that are passed to the Ingress EDS.

Table 7-47. Ingress Target Queue FCBPage Parameters

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

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 7-49* 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 7-48. ENQI Command Input

Name	Size	Operand Source		Description
		Direct	Indirect	
QueueClass	5	Imm16(4..0)	GPR(4..0)	*Table 7-49: Ingress-Queue Class Definition* on page 240
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 7-49. 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. 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 TDMU QCB queues (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 7-50. ENQCLR Command Input

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-51. ENQCLR Output

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

RELEASE_LABEL Command

This command releases the label in the Completion Unit for this frame. This command has no inputs.

Table 7-52. RELEASE_LABEL Output

Name	Size	Operand Source		Description
		Direct	Indirect	
Disp_Label	1	R		The Disp_Label register is reset on the execution of the release label command.

7.4.5 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 halfword data with a halfword checksum result.

7.4.5.1 Checksum Coprocessor Address Map

Table 7-53. 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 complement 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

7.4.5.2 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:

Command	Opcode	Description
GENGEN	0	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. For more information, see *GENGEN/GENGENX Commands* on page 243.
GENGENX	4	Generate Checksum with Cell Header Skip. For more information, see *GENGEN/GENGENX Commands* on page 243.
GENIP	1	Generate IP Checksum. For more information, see *GENIP/GENIPX Commands* on page 244.
GENIPX	5	Generate IP Checksum with Cell Header Skip. For more information, see *GENIP/GENIPX Commands* on page 244.
CHKGEN	2	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. For more information, see *CHKGEN/CHKGENX Commands* on page 244.
CHKGENX	6	Check Checksum with Cell Header Skip. For more information, see *CHKGEN/CHKGENX Commands* on page 244.
CHKIP	3	Check IP Checksum. For more information, see *CHKIP/CHKIPX Commands* on page 245.
CHKIPX	7	Check IP Checksum with Cell Header Skip. For more information, see *CHKIP/CHKIPX Commands* on page 245.

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 halfword 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 207 for more details of Cell Header Skip.

GENGEN/GENGENX Commands

Table 7-54. GENGEN/GENGENX Command Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
Load New Stake	1	Imm(6)	Imm(6)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear Accumulation	1	Imm(5)	Imm(5)	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(1:0,15:8)	Imm(1:0)
			GPR(23:16)
				The new stake value to be used for the checksum operation if the Load New Stake argument is '1'. The stake argument is comprised of two parts. The first (Imm1:0) is the Data Store coprocessor Array number. The second(Imm(15:8) or GPR(23:16)) is the byte offset within the array.

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 checksum

Table 7-55. GENGEN/GENGENX/GENIP/GENIPX Command Outputs

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-56. GENIP/GENIPX Command Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
Load New Stake	1	Imm(6)	Imm(6)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear Accumulation	1	Imm(5)	Imm(5)	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(4)	Imm(4)	IP Count clear control. 0 Continue with current count in the Checksum Length register 1 Clear counter and load with value in IP length field.

Stake Argument	10	Imm(1:0,15:8)	Imm(1:0)
			GPR(23:16)
				The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’. The stake argument is comprised of two parts. The first (Imm1:0) is the Data Store coprocessor Array number. The second(Imm(15:8) or GPR(23:16)) is the byte offset within the array.

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 7-55: GENGEN/GENGENX/GENIP/GENIPX Command Outputs* on page 243.

CHKGEN/CHKGENX Commands

Table 7-57. CHKGEN/CHKGENX Command Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
Load New Stake	1	Imm(6)	Imm(6)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.

Stake Argument	10	Imm(1:0,15:8)	Imm(1:0)
			GPR(23:16)
				The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’. The stake argument is comprised of two parts. The first (Imm1:0) is the Data Store coprocessor Array number. The second(Imm(15:8) or GPR(23:16)) is the byte offset within the array.

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 7-58. CHKGEN/CHKGENX/CHKIP/CHKIPX Command Outputs

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-59. CHKIP/CHKIPX Command Inputs

Name	Size	Operand Source		Description
		Direct	Indirect	
Load New Stake	1	Imm(6)	Imm(6)	Indicates: 1 Stake value is passed in the command. 0 Stake value is in ChkSum_Stake.
Clear Accumulation	1	Imm(5)	Imm(5)	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(4)	Imm(4)	IP Count clear control. 0 Continue with current count in the Checksum Length register. 1 Clear counter and load with value in IP length field.

Stake Argument	10	Imm(1:0,15:8)	Imm(1:0)
			GPR(23:16)
				The new stake value to be used for the checksum operation if the Load New Stake argument is ‘1’. The stake argument is comprised of two parts. The first (Imm1:0) is the Data Store coprocessor Array number. The second(Imm(15:8) or GPR(23:16)) is the byte offset within the array.

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 7-58: CHKGEN/CHKGENX/CHKIP/CHKIPX Command Outputs* on page 245.

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 halfword 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.

7.4.6 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.

7.4.6.1 String Copy Coprocessor Address Map

Table 7-60. 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).

7.4.6.2 String Copy Coprocessor Commands

Command	Opcode	Description
STRCOPY	0	For more information, see *StrCopy (String Copy) Command* on page 246

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 7-61. StrCopy Command Input

Name	Size	Operand Source		Description
		Direct	Indirect	
StrCpy_SAddr	14	R		Source Address. See *Table 7-60: String Copy Coprocessor Address Map* on page 246
StrCpy_DAddr	14	R		Destination Address. See *Table 7-60: String Copy Coprocessor Address Map* on page 246
NumBytes	8	Imm(7:0)	GPR(7:0)	Number of Bytes to transfer

Table 7-62. StrCopy Command Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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.

7.4.7 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.

7.4.7.1 Policy Coprocessor Address Map

Table 7-63. 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. Bit Description 19:10 Reserved. 9:0 Policy Control Block address.

7.4.7.2 Policy Coprocessor Commands

Command	Opcode	Description
POLACCESS	0	For more information, see *PolAccess (Access Policy Manager) Command* on page 247

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 7-64. PolAccess Input

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

Table 7-65. PolAccess Output

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

7.4.8 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 a thread 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.

7.4.8.1 Counter Coprocessor Address Map

Table 7-66. 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. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition would be used for this update.

7.4.8.2 Counter Coprocessor Commands

The Counter coprocessor provides the following commands:

Command	Opcode	Description
CtrInc	0	Counter Increment. Initiates a Modify and Increment command to the Counter Manager. For more information, see *CtrInc/CtrRInc (Counter Increment) Commands* on page 249.
CtrRInc	2	Reduced Counter Increment. Initiates a Modify and Increment command to the Counter Manager using an immediate field instead of loading the CtrControl scalar register. For more information, see *CtrInc/CtrRInc (Counter Increment) Commands* on page 249.
CtrAdd	1	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. For more information, see *CtrAdd/CtrRAdd (Counter Add) Commands* on page 250.
CtrRAdd	3	Reduced 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. This command uses an immediate field instead of loading the CtrControl scalar register. For more information, see *CtrAdd/CtrRAdd (Counter Add) Commands* on page 250
CtrRd	4	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. For more information, see *CtrRd (Counter Read) / CtrRdClr (Counter Read Clear) Command* on page 251.
CtrRdClr	5	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 to zero. For more information, see *CtrRd (Counter Read) / CtrRdClr (Counter Read Clear) Command* on page 251.
CtrWr15_0	6	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. For more information, see *CtrWr15_0 (Counter Write 15:0) / CtrWr31_16 (Counter Write 31:16) Command* on page 251.
CtrWr31_16	7	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. For more information, see *CtrWr15_0 (Counter Write 15:0) / CtrWr31_16 (Counter Write 31:16) Command* on page 251.

CtrInc/CtrRInc (Counter Increment) Commands

CtrInc and CtrRInc perform an accesses 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 7-67. Ctrinc Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	20		GPR(19..0)	Selects which set of counters to reference for this counter action.
CtrControl	12		R(11..0)	Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

Table 7-68. CtrRInc Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	20		GPR(19..0)	Selects which set of counters to reference for this counter action.
CtrControl	12	Imm (11:0)		Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

CtrAdd/CtrRAdd (Counter Add) Commands

CtrAdd and CtrRAdd perform accesses 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 7-69. CtrAdd Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	20		GPR (19..0)	Selects which set of counters to reference for this counter action.
CtrDataLo	16		R(15..0)	The value to be added to the counter.
CtrControl	12		R(11..0)	Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

Table 7-70. CtrRAdd Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	16		GPR (31:16)	Selects which set of counters to reference for this counter action.
CtrDataLo	16		GPR (15:0)	The value to be added to the counter.
CtrControl	12	Imm (11:0)		Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

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 7-71. CtrRd/CtrRdClr Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	20		GPR(19..0)	Selects which set of counters to reference for this counter action.
CtrControl	12		R(11..0)	Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

Table 7-72. CtrRd/CtrRdClr Output

Name	Size	Operand Source		Description
		Direct	Indirect	
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 7-73. CtrWr15_0/CtrWr31_16 Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SetIndex	20		GPR(19..0)	Selects which set of counters to reference for this counter action.
CtrDataLo	16		R(15..0)	The value to be written to the counter
CtrControl	12		R(11..0)	Counter Control. 11:8 Counter Number. Defines which counter in the Counter Set should be updated. 7:0 Counter Definition Table Index. Defines which counter definition should be used for this update.

7.4.9 Coprocessor Response Bus

The coprocessor response bus (CRB) is a bus interface and an internal coprocessor that enables an external coprocessor to write the CrbResults register and reset the busy bit of the CRB coprocessor. The 20-bit CrbResults register is written by an external device via the 14-bit coprocessor response bus. The 20 bits written into the CrbResults register by the external device are defined by the user. When the CrbResults register is written, the busy bit of the corresponding thread is automatically reset to indicate that the external device is done.

7.4.9.1 Coprocessor Response Bus Address Map

Table 7-74. Coprocessor Response Bus Coprocessor Address Map

Name	Register (Array) Number	Size (Bits)	Access	Description
CrbResults	x‘00’	20	R	Coprocessor response bus results. This field contains the 20 bits of data written to this thread’s CrbResults register from an external device. Data is written into the register via the 14-bit coprocessor response bus.

7.4.9.2 Coprocessor Response Bus Commands

Table 7-75. Coprocessor Response Bus Coprocessor Command Summary

Opcode	Command	Detail Section
0	CrbSetBusy	See *CrbSetBusy Command*.

CrbSetBusy Command

When the CrbSetBusy command is executed, the busy bit is set. The busy bit is reset when the CrbResults register is written by an external device. The CRB coprocessor does not indicate to the external device that the busy bit has been set. The external device must receive its data or commands via some other method.

Table 7-76. CrbSetBusy Output

Name	Size (Bits)	Operand Source		Description
		Direct	Indirect	
Return Code	1	CPEI signal		The operation always returns an OK ‘1’.
CrbResults	20	R		Coprocessor response bus results. This field contains the 20 bits of data written to this thread’s CrbResults register from an external device. Data is written into the register via the 14-bit coprocessor response bus.

7.4.9.3 14-bit Coprocessor Response Bus

From the 14-bit coprocessor response bus, twenty-four 20-bit registers can be written. One register is associated with each thread. These 20-bit registers are the CrbResults registers.

The 14-bit coprocessor response bus is a 14-bit input-only bus. These inputs are synchronous with the Z0 LU_Clk (see *Table 2-4: Z0 ZBT SRAM Interface Pins* on page 42). That is, the setup and hold requirements on these inputs are relative to the Z0 LU_Clk.

To write one of the 20-bit Crb0Results scalar registers, two consecutive clock cycles are used. The bit definitions for each clock cycle are shown in *Table 7-77*.

Table 7-77. 14-bit Response Bus Bit and Cycle Definition

Cycle	Response_Bus Bit	Description
0	13	Bit 13 is set to ‘1’. The first of two consecutive cycles is indicated by bit 13 being high.
	12:7	Thread number. The thread number indicates to which of the 24 CrbResults scalar registers the 20 bits of data are written. The thread number is a binary encoded value from 000000 to 011111.
	6:0	CrbResults (6:0).This cycle also contains seven of the 20 bits of data to be loaded into a CrbResults register.
1	13	Don’t care. This cycle always follows a cycle 0 and bit 13 is set to “don’t care.”
	12:0	CrbResults (19:7). This cycle contains the remaining 13 bits of data to be loaded into a CrbResults register.
Idle	13	Bit 13 is set to ‘0’. When the external device does not have any data to write into any of the 24 CrbResults registers, the external device must hold bit 13 low.
	12:0	Don’t care. During an idle cycle, these bits are set to “don’t care.”

7.4.10 Semaphore Coprocessor

The Semaphore coprocessor provides an interface for threads to the Semaphore Manager. The Semaphore coprocessor supports one outstanding coprocessor command per thread, and indicates a busy status until the command has been serviced.

7.4.10.1 Semaphore Coprocessor Commands

The Semaphore coprocessor provides the following commands:

Command	Opcode	Description
Semaphore Lock	0	Request to lock a semaphore. Parameters include: thread semaphore number, orderID, semaphore value, and TimeOutEnable. This command is complete when the semaphore is locked. The minimum time to complete a lock is five cycles. This represents the amount of time busy signal is asserted: five cycles if the lock is granted immediately, more if it is blocked.
Semaphore Unlock	1	Request to unlock a semaphore. Parameters include: thread semaphore number. This command is complete when the semaphore is unlocked. The time to complete an unlock is three cycles (the amount of time the busy signal is asserted).
Reservation Release	2	Request to remove a semaphore reservation from a queue. Parameters include: orderID. This command is complete when the reservation is released. The time to complete a release is three cycles (the amount of time the busy signal is asserted).

Note: The busy signal will not be asserted if a Lock no-op, Unlock no-op, or Reservation Release no-op is issued.

The following tables show the details of what is included in the commands:

Table 7-78. Semaphore Lock Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SemNr	1	Imm16(0)	Imm12(0)	Selects one of the two semaphores that can be owned by a thread.
TimeOutEnable	1	Imm16(1)	Imm12(1)	Enables timeout of command after one try. If a lock is not granted, the command returns with a KO. If a lock is granted, the command returns with an OK.
OrderID	2	Imm16(3..2)	Imm12(3..2)	00: Use Unordered Semaphore 01: Use Ordered Semaphore ID = 0 10: Use Ordered Semaphore ID = 1 11: no-op
4MSBAddressBits	4	Imm16(7..4)	Imm12(7..4)	These four bits are ORed to the upper 4 bits of address.
Address	32	Imm16(15..8)	GPR(31..0)	The Semaphore Value (in Direct Mode, the 8 Address bits are right-justified, the 4 MSBAddress Bits are ORed with x‘0’ and left-justified, and the remaining bits are 0s)
Note: There are no restrictions about how OrderID and SemNr are used in conjunction with each other. For example, although perfectly legal, there is no requirement that a lock request wishing to use Ordered Semaphore ID = 1 also request SemNr = 1. It is acceptable to request Ordered Semaphore ID = 1 and SemNr = 0, and vice versa.				

Table 7-79. Semaphore Unlock Input

Name	Size	Operand Source		Description
		Direct	Indirect	
SemNr	2	Imm16(1..0)	GPR(1..0)	Selects the semaphores that can be owned by a Thread. 00: no-op 01: Unlock Semaphore SemNr 0 10: Unlock Semaphore SemNr 1 11: Unlock Semaphore SemNr 0 AND 1

Table 7-80. Reservation Release Input

Name	Size	Operand Source		Description
		Direct	Indirect	
OrderID	2	Imm16(3..2)	GPR(1..0)	00: no-op 01: Release Ordered Semaphore ID 0 10: Release Ordered Semaphore ID 1 11: Release Ordered Semaphore ID 0 AND 1
Note: When using the Reservation Release instruction, the thread must wait for completion of this instruction before exiting. The Reservation Release command must be issued synchronously, or it must be issued asynchronously followed by a wait command to the Semaphore coprocessor.				

7.4.10.2 Error Conditions

These error conditions generate error interrupts on a per-thread basis:

1. Exit and Locked. If an Exit instruction is executed and the thread still has a locked semaphore, the semaphore manager will receive information from the DPPU indicating that a thread has exited. The semaphore manager unlocks the semaphore and then generates an error.

   If a thread issues an asynchronous lock request that goes pending and then exits before the request is completed, this error will be reported via the Semaphore Errors Register. This might cause this thread to hang during a subsequent dispatch, or cause another thread requiring the requested semaphore to hang.
2. Lock Same Sem Value. If a thread tries to lock a second semaphore with the same value as the first one it already has locked, the semaphore manager will not lock the semaphore and an error will be reported via the Semaphore Errors Register.
3. Lock Same Sem Number. If a thread tries to lock a semaphore number (0 or 1) that it already has locked (regardless of semaphore value), the semaphore manager will not grant the lock. It will unlock the semaphore that was in that position and will generate an error.
4. Queue Not Enabled. If a thread tries to lock an ordered semaphore, but the OrderEnable[ThreadNum] bit is not set, an error is generated. The semaphore manager will grant the lock anyway. The lock request will internally be made to look like an unordered lock request and will be processed as though it were an unordered lock request originally.
5. Label Released and Reservation Unused. When an Enqueue With Label or a Release Label instruction is executed and the thread still has an entry pending on the ordered semaphore queues ID = 0 or ID = 1 which it has not used, an error will be reported via the Semaphore Errors Register.

There is a precedence with regard to multiple errors present in one command (only errors 2, 3, and 4 described above can happen all in one request). The error precedence order for errors 2, 3, and 4 is as follows:

1. Error 3 (Lock Same Sem Number) will be detected and will prevent Errors 2 and 4 from being detected.
2. Error 4 (Queue Not Enabled) and Error 2 (Lock Same Sem Val) will both be detected if Error 3 is not present.

For example, a thread in its initial state might have no OrderID queues enabled and SemNr 0 locked with a Semaphore Value X. If that thread sends in a request for an ordered lock of SemNr 0 and Semaphore Value X, it results in error conditions 2, 3, and 4. However, the Semaphore Manager will detect one of the errors first (Error 3), generate the error interrupt and complete the request before detecting any of the other errors. On the other hand, assuming the same initial state as above, if a thread sends in a request for an ordered lock of SemNr 1 and Semaphore Value X, which results in errors 2 and 4, both errors will be detected and reported.

7.4.10.3 Software Use Models

A few rules should be followed when writing software that will use semaphores in order to prevent software lockups due to improper use of semaphores. These rules apply when the software is attempting to have two semaphores locked at the same time.

1. SemNr 0 and SemNr 1 should always be different “families” of semaphore values. In other words, a semaphore value locked in position SemNr 0 should never be locked in position SemNr 1. One way to accomplish this is to use the four most significant address bits to represent resources that are allowed to be locked in one of the two “families” of semaphore values.
2. SemNr 0 should always be locked first, followed by SemNr 1.
3. Ordered semaphore operations (ordered lock requests or reservation releases) must be completed before the first enqueue and are not permitted after that.
4. NP2G supports use of two semaphores when ordered semaphores are enabled. When using ordered semaphores, only one ordered semaphore should be used at a time and it should be locked first. A second semaphore can be locked at the same time if it is unordered.

7.5 Interrupts and Timers

The NP2G 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 7-84: Port Configuration Memory Content* on page 262.

7.5.1 Interrupts

The interrupt mechanism within the NP2G has several registers: the Interrupt Vector Registers 0-3, Interrupt Mask Register 0-3, Interrupt Target Register 0-3. and the Software Interrupt Registers.

7.5.1.1 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 as scalar registers (see *7.2.1.1 Core Language Processor Address Map* on page 169) or through the CAB interface to the master copy. The Interrupt Vector register is cleared when it is read from its Master Copy through the CAB interface but not when read as a scalar register.

7.5.1.2 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.

7.5.1.3 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.

7.5.1.4 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 addresses. 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.

7.5.2 Timers

The NP2G has four Timer Interrupt Counters that can be used to generate delayed interrupts to the EPC.

7.5.2.1 Timer Interrupt Counters

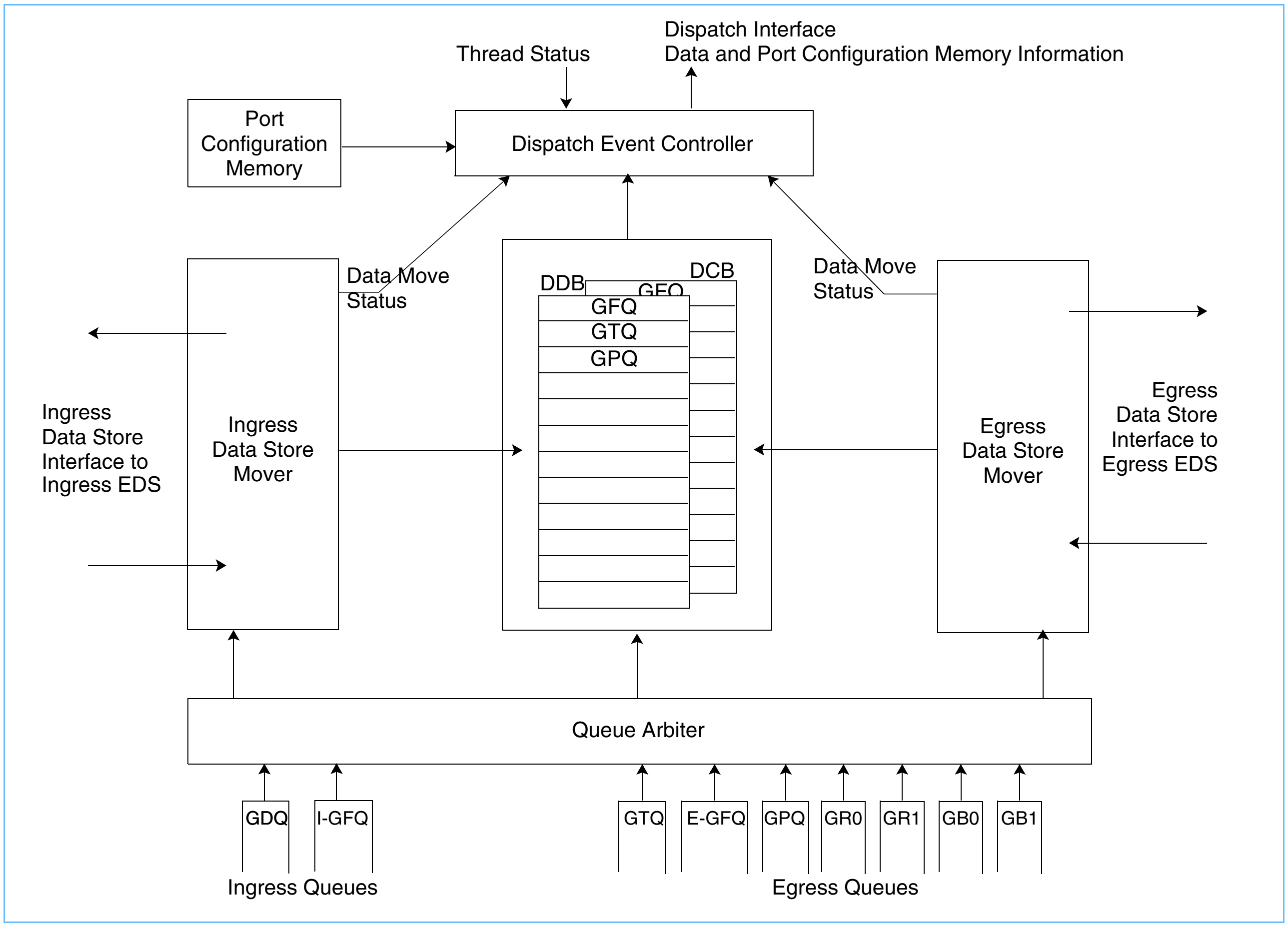
Timer Interrupt Counters 0-2 are 24 bits in length and decrement at 1 ms intervals. Timer Interrupt Counter 3 is 32 bits in length and decrements at 10 µs intervals. The Timer Interrupt Counters are activated by writing a non-zero value to the register. When the Timer decrements to a zero value it will schedule a timer task to be processed by the EPC

7.6 Dispatch Unit

The Dispatch Unit tracks thread usage and fetches initial data of frames prior to assigning a thread to a task. It also handles timers and interrupts by dispatching the work for these to an available thread.

The Dispatch Unit fetches frame data from the Ingress and Egress Data Stores for frame traffic work. The data is placed into the Dispatch Data Buffer (DDB), an array maintained by the Dispatch Unit. There are 24 entries. Each entry holds data for one frame dispatch. Three of the locations are fixed in use. The remaining 21 entries are used for work from the remaining egress frame queues, GR0, GR1, GB0, and GB1, and the ingress GDQ queues.

Figure 7-17. Dispatch Unit



The three fixed locations hold data for frame traffic from the ingress and egress Guided Frame Handler (GFQ) queues, the egress General Table Handler (GTQ) queue, and the request and egress only General PowerPC Handler (GPQ) queues.

The Dispatch Unit selects work from interrupts, timers, and frame queues. There are nine frame queues, four timer interrupts, and four hardware interrupts. If a queue is not empty and there is room in the DDB for data for a queue type, then the Dispatch Unit's Queue Arbiter considers the queue a candidate for selection via a priority, ingress/egress driven, round-robin process. See *Table 7-81: Priority Assignments for the Dispatch Unit Queue Arbiter* on page 260 for the queues and priority weights. The lower the priority number, the higher the priority selection weight.

Selection toggles between the ingress groups and egress groups. If there are no candidates in one group, then the other group may be selected during consecutive opportunities. In order to keep either the ingress or the egress work from taking up the entire DDB and thus starving the opposite group, a threshold is maintained for both. These thresholds are compared against the 21 entries in the DDB from the ingress and egress work groups. If a threshold is exceeded, then no further work for that group is allowed into the DDB. Simulation has shown that a value of 16 for each group's threshold maximizes the dispatch unit throughput and does not allow either group to starve from lack of processor resources.

Table 7-81. Priority Assignments for the Dispatch Unit Queue Arbiter

Queue	Priority
Ingress group	
I-GFQ	1
GDQ	2
Egress group	
E-GFQ	1
GTQ	2
GPQ	3
GR0	4
GR1	4
GB0	5
GB1	5

Once a frame has been selected by the queue arbiter, that frame's data is fetched into the DDB. Any relevant information about the frame's queue entry is fetched into the Dispatch Control Block (DCB). The amount of data fetched is dependent on the queue's port configuration memory contents. The port configuration memory is an array used by the dispatch unit that contains entries for all ingress ports, interrupts, and egress work queues.

The Dispatch Event Controller (DEC) schedules the dispatch of work to a thread. It monitors the status of all 24 threads in the EPC, the status of data movement into the DDB, and the status of the four hardware interrupts and four timers. Work from the GFQ, GTQ, and GPQ may only be processed by the GFH, GTH, and GPH-Req threads respectively. Threads that are allowed to handle interrupts and timers are configurable; they can be restricted to a single thread type, or can be processed by any thread. The DEC assures that there is a match between an available thread and the type of work to be performed. The DEC load balances threads on the available DPPUs and CLPs, keeping the maximum number of threads running in parallel in the EPC.

When a thread is available for work, and there is a match between the thread and ready work unit, and all the data has been moved for the work unit, then the work unit is dispatched to the thread for processing. The DEC provides the data fetched from the DDB and the contents of the Port Configuration Memory corresponding to the entry. When multiple units of work are ready to go that match available threads, the DEC toggles between ingress and egress work units.

There is no data movement for timers and interrupts. Only the contents of the Port Configuration Memory entry are passed to the thread.

7.6.1 Port Configuration Memory

Table 7-84: Port Configuration Memory Content on page 262 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.

7.6.1.1 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 7-82*.

Table 7-82. 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 7-83. Relationship Between SP Field, Queue, and Port Configuration Memory Index

SP field in FCB1	Queue	Port Configuration Memory Index
0, 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, 39 (denotes physical port)	GDQ	0, 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, 39
0, 2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31, 34, 35, 38, 39 (denotes physical port)	I-GFQ	42
40 (denotes wrap port)	GDQ	40
40 (denotes wrap port)	I-GFQ	43

7.6.2 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 7-84*.

Table 7-84. Port Configuration Memory Content

Field Name	Bits	Description
	127 .. 25	For Software use
IPv4 Forwarding Enabled	24	Use of this bit is enabled by configuration of the Enhanced Classification Enable register (see *Section 13.14.10* on page 467): 0 IPv4 forwarding disabled 1 IPv4 forwarding enabled
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: 00 four quadwords 01 one quadword 10 two quadwords 11 three 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.

7.6.3 Completion Unit

The Completion Unit provides the interface between the EPC and the Ingress Enqueuer/Dequeuer/Scheduler (Ingress EDS) and Egress EDS. During dispatch, the thread is identified and a label for the frame being dispatched is created. Frames from the same communication flow receive the same label. Using these labels and thread IDs, the Completion Unit maintains frame order within communication flows. For example, the Completion Unit blocks completion of a thread until all threads with the same label dispatched before that thread have been enqueued and completed.

Frames dispatched without labels must be enqueued without labels. Frames enqueued without labels are not blocked by other threads pending completion (that is, they are completed as soon as resources are available). Because of this, frames without labels are not guaranteed to stay in order.

7.7 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.

7.7.1 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.

7.7.1.1 Ingress Classification Input

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

- Port Configuration Memory Table Entry (*Table 7-84* on page 262). 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 7-85. 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

Note: Within each group of interface types (Ethernet, Point to Point, or Service Access Point), the protocol identifier values must be unique. This restriction includes setting up a programmable protocol identifier to be equal to a fixed protocol identifier.
The fixed protocol identifier for PPP control frames (x‘2500 00F00’) creates a conflict with the programmable protocol identifier. This conflict is resolved in the following ways:

- A complete (all bits) protocol match is selected over a PPP control match.

7.7.1.2 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 7-84: Port Configuration Memory Content* on page 262) 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 7-86: HCCIA Table* on page 266.

Table 7-86. HCCIA Table (Page 1 of 2)

HCCIA CAB Address	Bits	Classification	Notes
x‘2500 0400’ x‘2500 05F0’	16	Egress Locations (see section *7.7.2.1 Egress Classification Input* on page 269)	
x‘2500 0600’	16	Ethernet Protocol 0 classification with no 802.1Q VLAN	1
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	1
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	1
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	1
x‘2500 0770’	16	Point to Point Control Frame	
x'2500 0780'	16	Ethernet IPv4 multicast with no 802.1Q VLAN	2, 4
x'2500 0790'	16	Ethernet IPv4 multicast with 802.1Q VLAN	2, 4
x'2500 07A0'	16	Point to point IPv4 multicast	2, 4

Notes:

1. In NP2G, when enabled by configuration of the Enhanced Configuration register (see *Section 13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)* on page 467), this entry is used for IP4v exceptions. An exception entry is used in the following circumstances:
   - The PCT entry for IPv4 forwarding (bit 24) is ‘0’.
   - The version and IP Header Length fields of the IPv4 header is not x‘45’.
   - The first four bits of the IP DA is greater than x‘E’.
2. IPv4 multicast is determined by examination of the IP DA, where the first four bits of the address are x‘E’.
3. IPv4 unicast is determined by examination of the IP DA, where the first four bits of the address is less than x‘E’.
4. Available in NP2G when enabled by configuration of the Enhanced Configuration register (see *Section 13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)* on page 467).

Table 7-86. HCCIA Table (Page 2 of 2)

HCCIA CAB Address	Bits	Classification	Notes
x'2500 07B0'	16	Reserved	4
x'2500 07C0'	16	Ethernet IPv4 multicast with no 802.1Q VLAN	3, 4
x'2500 07D0'	16	Ethernet IPv4 multicast with 802.1Q VLAN	3, 4
x'2500 07E0'	16	Point to point IPv4 multicast	3, 4
x'2500 07F0'	16	Reserved	4

Notes:

1. In NP2G, when enabled by configuration of the Enhanced Configuration register (see *Section 13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)* on page 467), this entry is used for IP4v exceptions. An exception entry is used in the following circumstances:
   - The PCT entry for IPv4 forwarding (bit 24) is ‘0’.
   - The version and IP Header Length fields of the IPv4 header is not x‘45’.
   - The first four bits of the IP DA is greater than x‘E’.
2. IPv4 multicast is determined by examination of the IP DA, where the first four bits of the address are x‘E’.
3. IPv4 unicast is determined by examination of the IP DA, where the first four bits of the address is less than x‘E’.
4. Available in NP2G when enabled by configuration of the Enhanced Configuration register (see *Section 13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)* on page 467).

Table 7-87. 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

- 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.

  The fields in the frame data that correspond to Data Store coprocessor Register Settings (see *Table 7-87*) 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 7-88. General Purpose Register Bit Definitions for Ingress Classification Flags

Bit	Definition	Notes
Bit 15	Protocol 7 detected	1
Bit 14	Protocol 6 detected	1
Bit 13	Protocol 5 detected	1
Bit 12	Protocol 4 detected	1
Bit 11	Protocol 3 detected	1
Bit 10	Protocol 2 detected	1
Bit 9	Protocol 1 detected	1
Bit 8	Protocol 0 detected	1
Bit 7	IPv4 TTL check. This bit is set to ‘1’ when the value in the TTL field ≤ 1.	2
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	

Notes:

1. In most cases only one bit in the range 15:8 is set. The exception is for a PPP interface where there is a match on a unique protocol identifier and the PPP Control type identifier.

2. Available in NP2G when enabled by configuration of the Enhanced Configuration register (see *Section 13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)* on page 467). When not available or disabled by configuration, this bit is set to ‘0’.

- 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 7-89. 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

7.7.2 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.

7.7.2.1 Egress Classification Input

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

- Port Configuration Memory Table Entry (*Table 7-84* on page 262). The Hardware Classifier uses the following fields:
  - CodeEntryPoint- the default starting instruction address
  - HardwareAssistEnabled
  - CULabGenEnabled
- Frame Data. One quadword must be dispatched (per frame) for egress classification.

7.7.2.2 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 7-84: Port Configuration Memory Content* on page 262) is used as the Starting Instruction Address and is passed to the thread. The address into HCCIA is given by the UC field and by the FHF field in the frame header:

Table 7-90. 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 7-10: Egress Frames DataPool Quadword Addresses* on page 208

- Classification Flags:

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

Table 7-91. General Purpose Register 1 Bit Definitions for Egress Classification Flags

Bit	Definition
Bit 15:1	Reserved
Bit 0	A link pointer error was found during Dispatch Unit access of the Egress data store. Recommended action is to discard the frame.

7.7.3 Completion Unit Label Generation

If label generation is enabled for a given dispatch the following table defines the construction of the 28-bit label to be passed to the Completion Unit.

Table 7-92. Completion Unit Label

	Bit28	27..25	24..0	(Bits)
Ingress Ethernet Frame	1	000	Hashed (MAC SA/DA) SP	24..6 5..0
PPP Ingress Frame	1	001	0 Protocol field SP	24..22 21..6 5..0
Ingress Unknown Protocol	1	010	0 SP	24..6 5..0
Egress UC Frame	0	000	0 LID	24..21 20..0
Egress MC Frame	0	001	MID 0 BranchNr	24..8 7 6..0
Reserved	0	010	0	
Reserved	0	011	0	
Reserved	0	100	0	
Reserved	0	101	0	

- Enabling Label generation is programmable in the Port Configuration Memory. If not enabled, the completion unit will not assure frame ordering for this frame. Label generation should not be used for guided frames, interrupts, and timers.
- When BranchNr is used to form the label, it is initialized to 0 by the Hardware Classifier. (It is incremented by the completion unit during multicast enqueues).

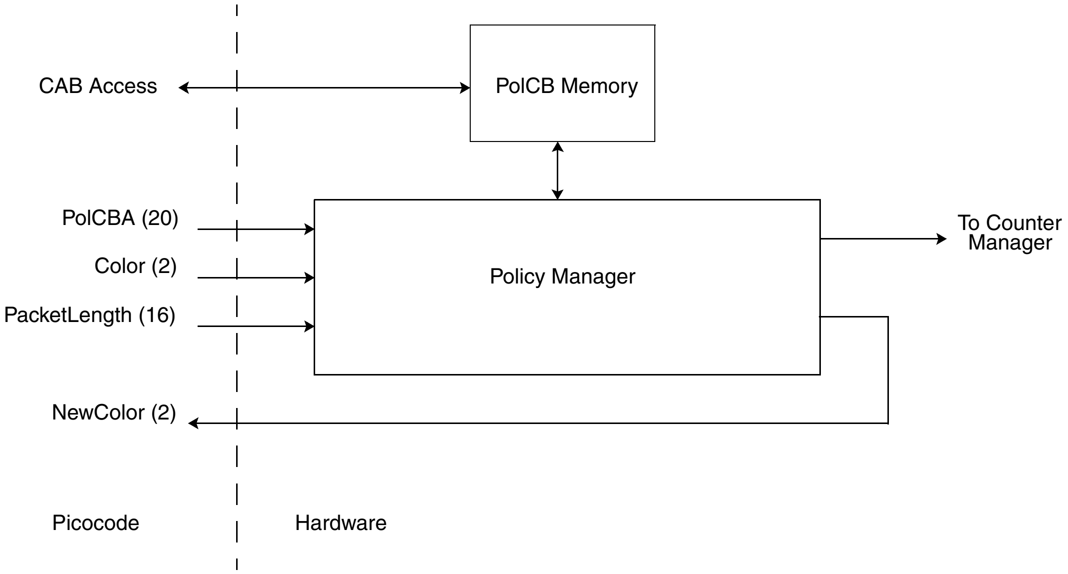
7.8 Policy Manager

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 (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 Manager 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 7-18. 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 7-93: PolCB Field Definitions* on page 273.

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. These are re-encoded as received in the DS Byte as 00 = green, 01 = yellow, 10 = red. Note that the encoding used does not conform to the RFC 2597.
- 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. The Policy Manager can perform these operations once every 15 core clock cycles. Picocode might use this information as follows:

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

Table 7-93. PolCB Field Definitions (Page 1 of 2)

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 165/150 ns. 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 165 ns 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 165 ns, when set to 0 the tick is 150 ns. 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 100 Kbps through 3 Gbps.
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 ($11/\overline{10}$). When set to 1 the tick is 165 ns, when set to 0 the tick is 150 ns. 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 100 Kbps through 3 Gbps.

Table 7-93. PolCB Field Definitions (Page 2 of 2)

Field	Size	Description
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. **Note:** For the two rate algorithms, C_BurstSize must be greater than 0. It is recommended that it be larger than 1.5 times the MTU size 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. When EP_Burstsize is larger than 0, the value loaded must be the sum of the CBS and EBS parameters described in RFC 2697. **Note:** For the two rate algorithms, EP_BurstSize must be greater than 0. It is recommended that it be larger than 1.5 times the MTU size 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.

7.9 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 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 Kb, 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 six DPPUs, the Policy Manager, Ingress and Egress Flow Control
- Five Independent Counter Storage locations

Figure 7-19. Counter Manager Block Diagram

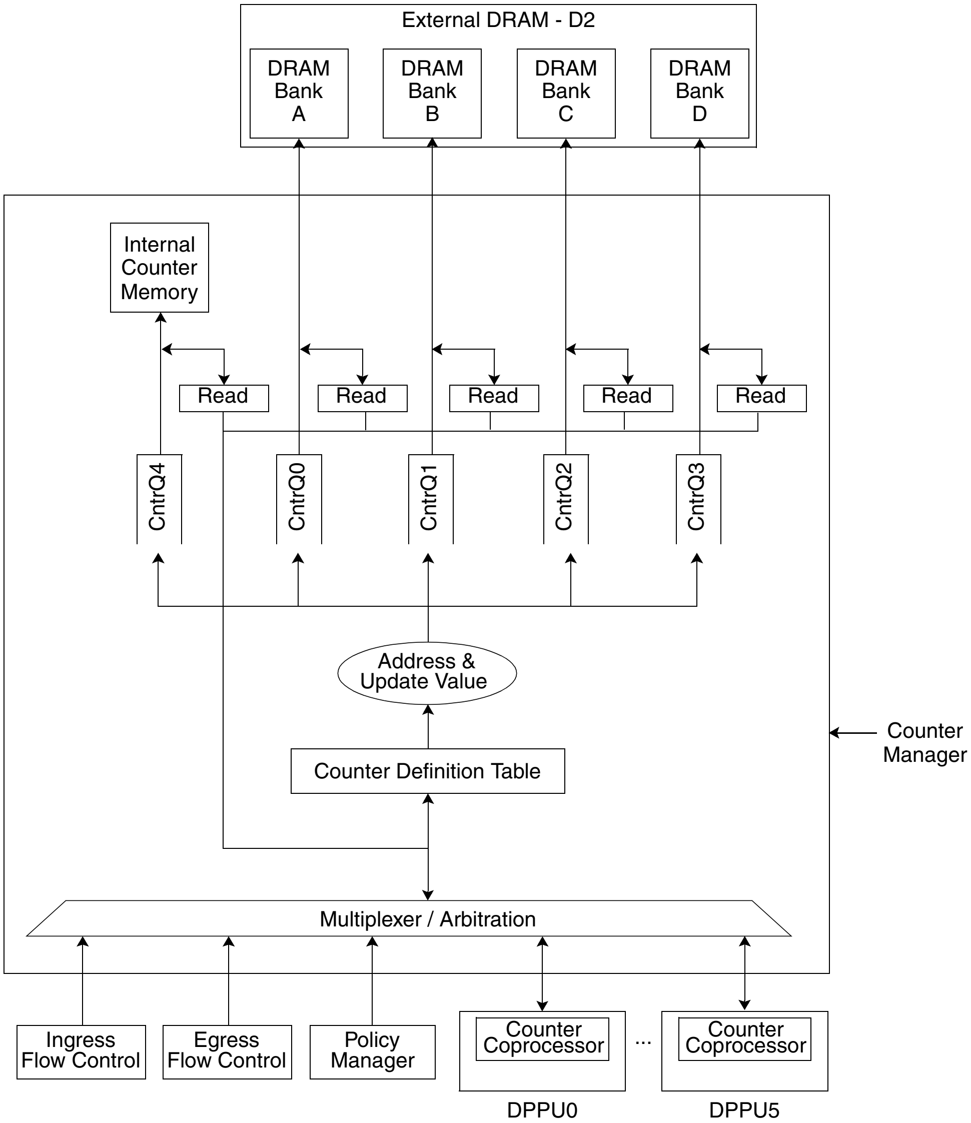


Table 7-94. Counter Manager Components

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 7-95. 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 7-96. 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.

7.9.1 Counter Manager Usage

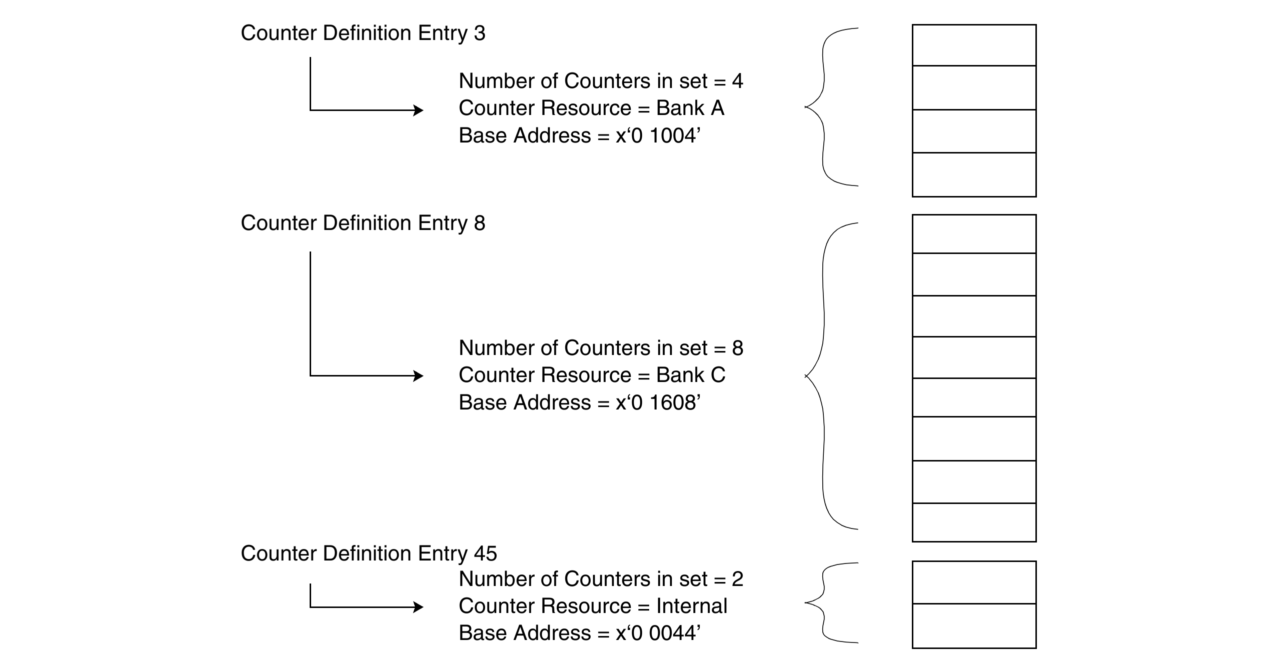
The Counter Manager manages various counters for the EPC. The EPC, Policy Manager, and the Ingress and Egress Flow Control have the ability to update these counters. The picocode accesses these counters for statistics, flow control actions, and policy decisions. Before a counter can be used, its counter 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 7-97* on page 278.

Table 7-97. 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. Counter addresses are based on a 64-bit word, however, all address bits are not used for all counter resource locations and when a 32-bit counter is indicated (bit 28 is set to 0), address bit 0 indicates the counter location within the 64-bit word (0 = bits 31:0, 1 = bits 63:32). The following illustrates the use of the base address bits. 32-bit counter / Counter Resource Location / 64-bit word address ---------- / ------------------------- / -------------- Yes / '000' - '011' / 19:1 Yes / '100' / 11:1 No / '000' - '011' / 19:0 No / '100' / 10:0

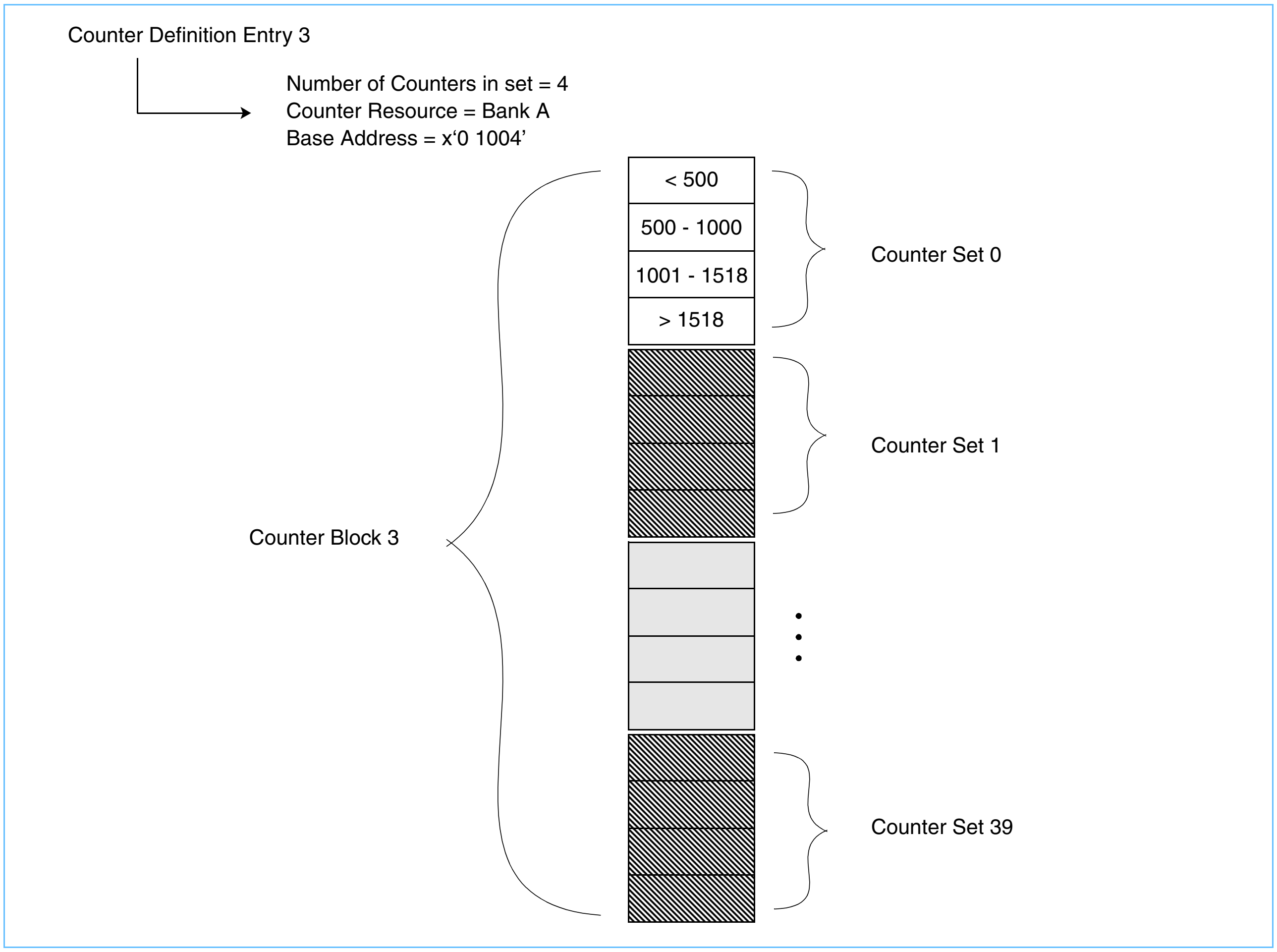
The Counter Definition Entry describes a set of counters that have similar characteristics and are referenced from the picocode as a single group. *Figure 7-20* on page 279 shows several counter definition examples.

Figure 7-20. Counter Definition Entry


Counter Definition Entry 3
Number of Counters in set = 4
Counter Resource = Bank A
Base Address = x‘0 1004’
Counter Definition Entry 8
Number of Counters in set = 8
Counter Resource = Bank C
Base Address = x‘0 1608’
Counter Definition Entry 45
Number of Counters in set = 2
Counter Resource = Internal
Base Address = x‘0 0044’

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, that is, 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 7-21* on page 280 shows an example of counter blocks and sets.

Figure 7-21. Counter Blocks and Sets



To reference the correct counter, the requestor must pass the Counter Manager several parameters. These parameters are described in *Table 7-98* on page 281.

Table 7-98. 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; all other counter bits are set to zero 111 Write bits 31:16 of counter; all other counter bits are set to zero 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 location of the counter and its size determine how many address bits are used by the Counter Manager as seen in *Table 7-99*.

Table 7-99. Counter Manager use of Address Bits

Memory Location	Counter Size	Number of Counters stored at Single Memory Address	Number of Address bits used	Total Counters Possible
Internal	32	2	11 (10:0) where bit 0 selects upper or lower 32 bits	1K loc * 2 per = 2K
	64, 24/40	1	10 (9:0)	1K loc * 1 per = 1K
DRAM Banks (A, B, C, or D)	32	2	20 (19:0) where bit 0 selects upper or lower 32 bits	512K loc * 2 per * 4 banks = 4M
	64, 24/40	1	20 (19:0)	1M loc * 1 per * 4 banks = 4M

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 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.

The Counter Manager arbitrates for a new counter action at a rate of one each 15 ns. 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), and one to four counters (one per bank) will be updated during this time. If the Counter Manager is updating some (but not all) DRAM banks, the banks not being updated by the Counter Manager will have their last location (address = all 1's) written during the DRAM write window. Therefore, the last location of each DRAM bank must be reserved for this use and cannot be assigned as a counter location.

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 (this feature is provided to assist in chaining unused counters into a free list).

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 Kb 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.

7.10 Semaphore Manager

The semaphore manager is located within the EPC and is controlled by a thread through a semaphore coprocessor.

A semaphore is a mechanism for acquiring ownership or “locking down” an entity. A semaphore is a 32-bit value. Once a thread has exclusive ownership of a semaphore, it is guaranteed that there are no other threads that own a semaphore with the same value (though there may be other threads owning semaphores with different values). Thus, other threads that also want ownership of a semaphore with the same value are blocked until the semaphore is unlocked.

It is upon the programmer to attach a meaning to a semaphore. That is, the semaphore manager does not know what a semaphore represents - it is just a string of 32 bits. Semaphores can be seen as having a 32-bit address space and the programmer can map this to anything, for example, the Tree Search Memory, the data store, or the Embedded PowerPC. For example, a tree search memory can have the upper four bits of the semaphore value set to ‘0000’, an ingress data store address can have the upper four bits of the semaphore value set to ‘0001’, and the embedded PowerPC mailbox may have the upper four bits set to ‘1111’.

When configured for ordered semaphores, a thread can have two semaphores locked at a time, but only one of them can be ordered.

- Unordered Semaphores - When multiple threads request a semaphore, the semaphore manager will grant the request through a round-robin fairness algorithm. A thread can lock-and-unlock an unordered semaphore as often as it wants. For example, it can lock-and-unlock, execute some code, lock again, execute more code, unlock, etc.
- Ordered Semaphores - When multiple threads request a semaphore, the semaphore manager will grant the request in the order of frame dispatch for a given flow. In other words, when a thread is processing a frame of a certain flow that has been dispatched before any other frames of the same flow, then it is guaranteed by the semaphore manager that this thread will get a certain semaphore value before any other threads that are processing frames of the same flow. The dispatch order will be maintained by placing a “reservation” into a queue. Only semaphore requests in which their reservation is on the top of the queue will be serviced. It will be the responsibility of the picocode to release a reservation if it isn’t needed or a deadlock situation will occur.

To use ordered semaphores, they must be enabled. This is done by writing the ordered semaphore enable register in the hardware classifier. There is one bit for the semaphore 0 ID queue, and one bit for the semaphore 1 ID queue. At dispatch time, the hardware classifier may assign a Completion Unit label to a frame as per the Port Configuration table.[1] If a Completion Unit label is assigned, the hardware classifier will examine the ordered semaphore enable register to see if ordered semaphores are enabled as well. If Completion Unit label generation is not enabled, ordered semaphores will not be enabled for this dispatch. The Completion Unit and the semaphore ordering queues use the same label. If a Completion Unit label is released (via an enqueue with label or release label command), the semaphore order label must first have either been used or released, otherwise an error will occur.

A thread can lock-and-unlock an ordered semaphore up to two times. During a request, the thread must specify the OrderID, which is 0 or 1. An order queue can only be used once, with two order queues implemented for a total of two ordered semaphores per thread. In other words, a thread can only lock OrderID 0 once, and OrderID 1 once.

[1] When the CULabGenEnabled field of the Port Configuration Table (see *Table 7-82* on page 261) is set to ‘1’.

Each thread can have ownership of up to two semaphores simultaneously. When using two semaphores, the programmer must take great care to avoid deadlock situations (semaphores can only be requested one by one). For example, a semaphore lock *must* be followed at some point by a semaphore unlock or a deadlock will occur.

The semaphore manager employs a pending concept to ordered semaphore lock requests, which enables it to look past the head of the completion unit queues. The basic idea is if a thread is requesting an ordered lock for a particular semaphore value that is already locked, it will be moved into a pending state, and it's request will not be considered complete. At this point, the threads that are behind it in the flow queues can now be considered for locking as long as their semaphore values are different. Once the original semaphore value that was locked is unlocked, then the pending thread's semaphore value will lock and that thread's request will be completed. It is important to note that only one thread per semaphore value will go into the pending state.

The semaphore manager will process the three semaphore coprocessor command types in parallel:

- Lock commands for ordered semaphores at the head of the queue and lock commands for unordered semaphores will be evaluated at the same priority level and a winner will be chosen using a round-robin fairness algorithm.
  - Lock commands will be evaluated in parallel with Unlock and Reservation Release commands.
- Unlock commands
  - Unlock commands will be evaluated in parallel with Lock and Reservation Release commands.
- Reservation Release commands
  - Reservation Release commands will be evaluated in parallel with Lock and Unlock commands.

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, D2, and D3).

8.1.1 Addressing Control Store (CS)

References to the control store use a 26-bit address as shown in *Table 8-1*. Each address contains a memory ID, a bank number, and address offset.

Table 8-1. 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 Number	Offset
Virtual Bank Address		

Table 8-2 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 434).

Table 8-2. CS Address Map and Use (Page 1 of 2)

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 1M x 36	20	Fast PSCB
	01-11	Reserved			
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			

Table 8-2. CS Address Map and Use (Page 2 of 2)

Memory ID	Bank No.	Memory Name	Type	Offset Width (bits)	Recommended Use
1000	00-11	D0 banks A-D	DDR DRAM D0_Width = 0 8 MB 16 MB 32 MB D0_Width = 1 16 MB 32 MB 64MB	18-20	Single (D0_width = 0) or double wide (D0_width = 1) configuration. Leaf Pages
1001			DDR DRAM 8 MB 16 MB 32 MB	18-20	Reserved
1010	00-11	D2 banks A-D			Leaf or Counter pages
1011	00-11	D3 banks A-D			DT entries and PSCB entries
1100	00-11	D6 banks A-D	DDR DRAM 8 MB 16 MB 32 MB 64MB 128 MB	20	PowerPC external memory
1101					
1110					
1111					

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

8.1.2 D6 Control Store.

The NP2G supports the following DDR DRAM types for the D6 control store:

DDR DRAM	Total Memory Space (MB)
4 x 1M x 16 × 1 chip	8
4 x 2M x 16 × 1 chip	16
4 x 4M x 4 × 4 chips	32
4 x 4M x 16 × 1 chip	32
4 x 8M x 4 × 4 chips	64
4 x 16M x 4 × 4 chips	128

The D6 control store can be accessed either via the TSE or by the embedded PowerPC's Processor Local Bus (PLB) (see *10.2 Processor Local Bus and Device Control Register Buses* on page 362).

8.1.3 Logical Memory Views of D6

Using the PLB, the embedded PowerPC or a PCI attached host views a flat, contiguous address space addressable on byte boundaries.

Access via the TSE uses the Control Store address mapping described in *8.1.1 Addressing Control Store (CS)* on page 285, and is composed of a Memory ID, Bank Number and offset. The offset limits addressing capability to 8-byte boundaries within the DDR DRAM.

Table 8-3 illustrates the PLB to D6 control store address translation. The PLB address is shown within the body of the table and the D6 control store address is formed using the Memory ID, Bank Number, and Offset.

Table 8-3. PLB and D6 Control Store Addressing

Memory ID (4 bits)	Bank A (‘00’)	Bank B (‘01’)	Bank C (‘10’)	Bank C (‘11’)	Offset (20 bits)
‘1100’	0-7	8-x‘F	x‘10-x‘17	x‘18 - x‘1F	x‘00 0000’
	x‘20-x‘27	x‘28-x‘2F	x‘30-x‘37	x‘38-x‘3F	x‘00 0001’
	x‘07F FFE0 - x‘07F FFE7	x‘07F FFE8 - x‘07F FFEF	x‘07F FFF0 - x‘07F FFF7	x‘07F FFF8 - x‘07F FFFF (8MB-1)	x‘03 FFFF’
	x‘0FF FFE0 - x‘0FF FFE7	x‘0FF FFE8 - x‘0FF FFEF	x‘0FF FFF0 - x‘0FF FFF7	x‘0FF FFF8 - x‘0FF FFFF (16MB-1)	x‘07 FFFF’
	x‘1FF FFE0 - x‘1FF FFE7	x‘1FF FFE8 - x‘1FF FFEF	x‘1FF FFF0 - x‘1FF FFF7	x‘1FF FFF8 - x‘1FF FFFF (32MB-1)	x‘0F FFFF’
'1101'	x‘200 0000 - x‘200 0007	x‘200 0008 - x‘200 000F	x‘200 0010 - x‘200 0017	x‘200 0018 - x‘200 001F	x‘00 0000’
	x‘3FF FFE0 - x‘3FF FFE7	x‘3FF FFE8 - x‘3FF FFEF	x‘2FF FFF0 - x‘2FF FFF7	x‘3FF FFF8 - x‘3FF FFFF (64MB-1)	x‘0F FFFF’
	x‘400 0000 - x‘400 0007	x‘400 0008 - x‘400 000F	x‘400 0010 - x‘400 0017	x‘400 0018 - x‘400 001F	x‘00 0000’
	x‘5FF FFE0 - x‘5FF FFE7	x‘5FF FFE8 - x‘5FF FFEF	x‘5FF FFF0 - x‘5FF FFF7	x‘5FF FFF8 - x‘5FF FFFF (96MB-1)	x‘0F FFFF’
'1111'	x‘600 0000 - x‘600 0007	x‘600 0008 - x‘600 000F	x‘600 0010 - x‘600 0017	x‘600 0018 - x‘600 001F	x‘00 0000’
	x‘7FF FFE0 - x‘7FF FFE7	x‘7FF FFE8 - x‘7FF FFEF	x‘7FF FFF0 - x‘7FF FFF7	x‘7FF FFF8 - x‘7FF FFFF (128MB-1)	x‘0F FFFF’

D6 control stores implemented in DDR DRAM of up to 32 MB in size are accessed by the TSE using only Memory ID '1100'. Larger D6 control stores require Memory IDs of '1101' (up to 64 MB), '1110' (up to 96 MB) and '1111' (up to 128 MB).

A method to share memory objects between the embedded PowerPC or attached PCI host using the PLB, and picocode running on the NP2G using TSE instructions is to specify the object (as viewed by the TSE) with a height of one and a width of four. This allows the TSE access to all of the D6 control store, on 32-byte boundaries, and is consistent with the PLB's flat contiguous address space view. Successive objects with a height of one and a width of four are accessed by incrementing the TSE address until a 32MB boundary is reached. At the boundary it is necessary to increment the Memory ID to address the next successive object.

8.1.4 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.
- PSCBs can be stored only in H1, Z0, or D3.
- Leaf pages can be stored only in H0, D0, and D2.

8.1.5 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 8-5: Height, Width, and Offset Restrictions for TSE Objects on page 290* and *Table 8-17. LUDefTable Entry Definitions* on page 307 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 8-4. DTEntry, PSCB, and Leaf Shaping

Object	Shape
FM DTEntry/PSCB	Always has a shape of height = 1, width = 1.
SMT DTEntry/PSCB	Always has a shape of height = 1, width = 1.
LPM DTEntry/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 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. For the accesses with height or width of more one, other requesters are not blocked. Restrictions to height, width, and offset are given in *Table 8-5: Height, Width, and Offset Restrictions for TSE Objects on page 290*. *Table 8-17: LUDefTable Entry Definitions* on page 307 specifies the leaf and PSCB shapes.

Figure 8-1. Example Shaping Dimensions on page 289 illustrates some example placement of objects with different shapes in control store. An object cannot span more than one memory:

Figure 8-1. Example Shaping Dimensions

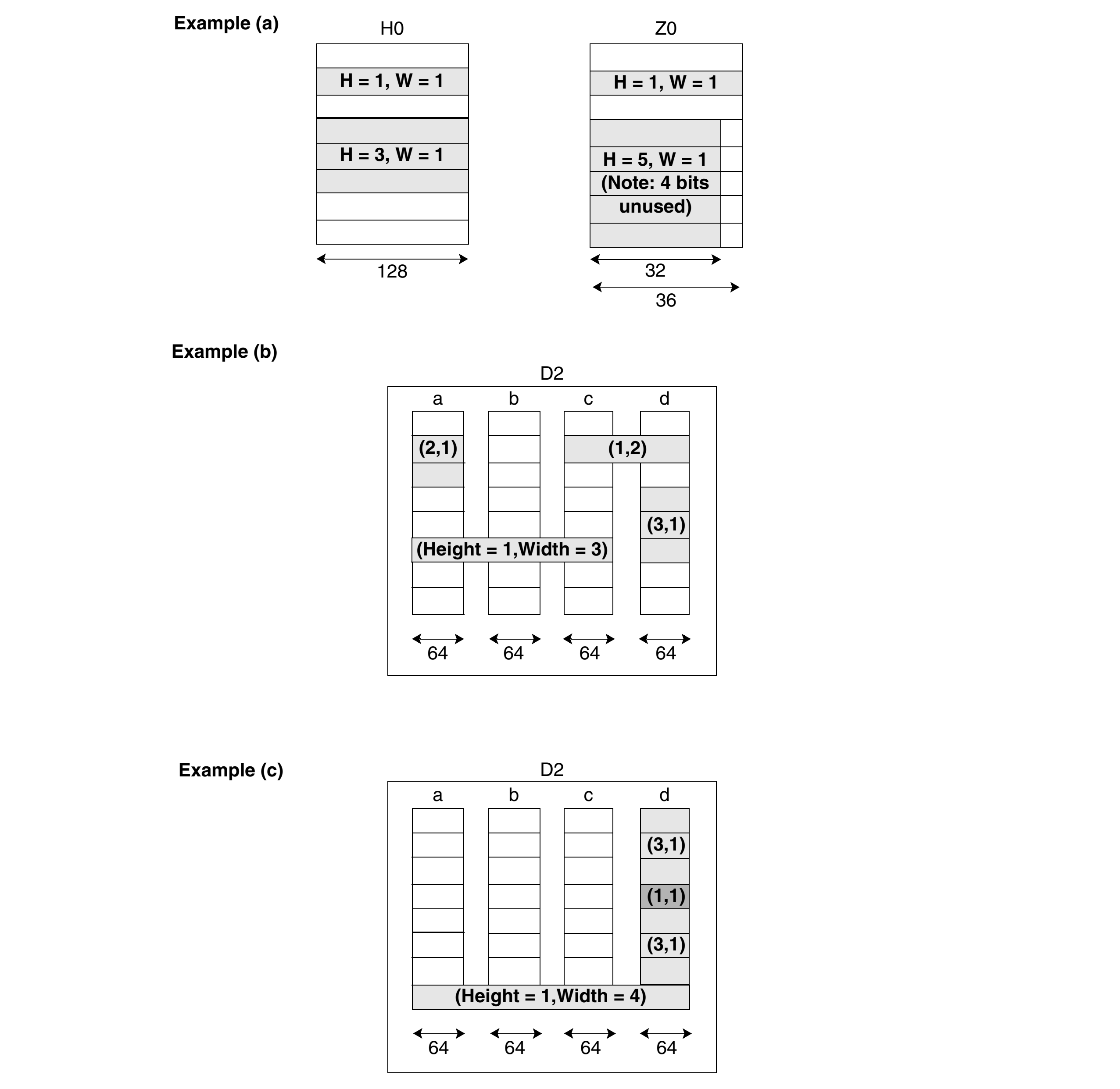


Table 8-5. Height, Width, and Offset Restrictions for TSE Objects

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

8.1.6 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 (TSM_Illegal_Memory_Access). The exception flag can be set to cause an interrupt by setting bit 2 of interrupt mask 3 to ‘1’ (TSM_Illegal_Memory_Access). For debugging purposes, an exception can switch a programmable set of threads into single-step mode.

8.1.7 Memory Range Checking (Address Bounds Check)

Memory range checking can flag access to a programmable range within Min_addr_Bounds and Max_addr_Bounds for read, write or both using Addr_Bounds_Cntl. Min_addr_Bounds. Memory range checking can be performed for any TSE Control Store accesses only. When memory range checking is enabled, any Control Store read, write, or read/write address falling within or at the boundary of the defined 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’ (TSM_Addr_Range_violation). For debug purpose, Bounds Violation Register (Bounds_vio), indicates which GxH has caused an address bounds exception.

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 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 8-6. 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	2-24	Pattern to be compared with HashedKey. Always present. Length given by P1P2_max_size from *Table 8-17: LUDefTable Entry Definitions* on page 307. For FM: Unused bit between (p1p2_max_size - Hashedkey_length) must be initialize to zero. For LPM: Unused bits between (p1p2_max_size - Prefix_Len), do not have to be initialized and can be used by user data since these bits do not take part in comparsion.
UserData	Variable	Under picocode control. For example, field can include one or more counters.

Table 8-7. 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	4-48	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 8-17: LUDefTable Entry Definitions* on page 307. For SMT: Unused bit between (2 * p1p2_max_size - 2 * Hashedkey_length) must be initialized to zero.
UserData	Variable	Under picocode control. For example, field can include one or more counters.

Table 8-8. 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 307.
TSEDPA	4	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 *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary. Use only values of x‘8’, x‘A’, x‘C’, and x‘E’.
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 293 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 hash algorithm specified by the Hash_type field 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 inserted after the input key has been hashed. This occurs immediately 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.

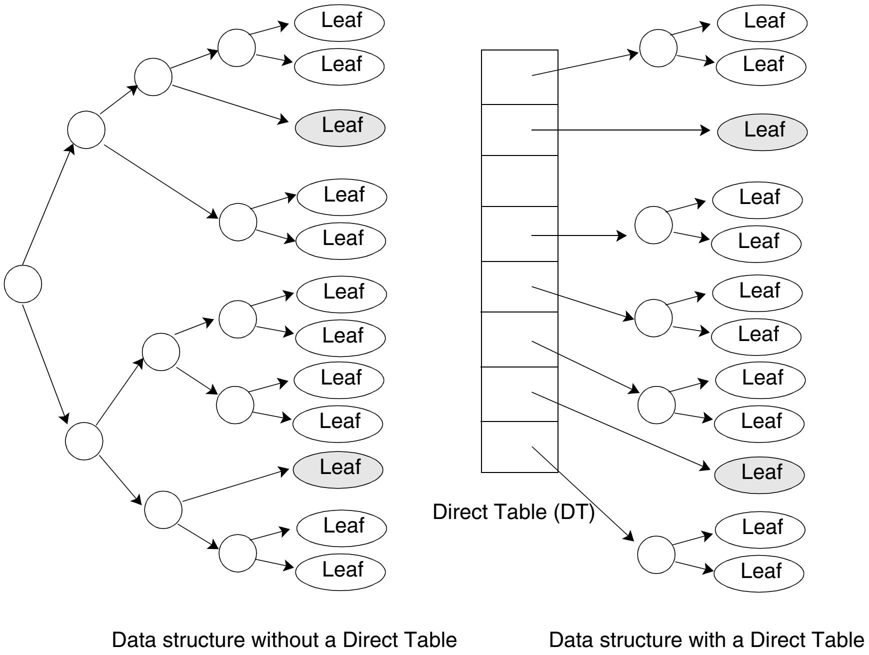
8.2.2 Input Key and Color Register for SMT Trees

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 8-2. Effects of Using a Direct Table


Leaf
Direct Table (DT)
Data structure without a Direct Table
Data structure with a Direct Table

8.2.3.1 Pattern Search Control Blocks (PSCB)

A search begins when 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. For SMT, 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 Cascade/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 pointer to 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.

Cascades/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. Cascading/caching of more than one LUDefTable entry is supported but the last Ludef-Table entry must have the cache enable bit set to '0' (disabled).

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 8-9*. Each register is divided into two sections, one for searches using TSE 0 coprocessor and the other for searches using TSE 1 coprocessor. There are two copies of these registers for each TSE 0 and TSE 1 resource. CacheFlags represents the results for the first Cached Ludef entry when multiple search chaining is used.

Table 8-9. 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 8-10*.

Table 8-10. Search Output Parameters

Parameter		Description
OK/KO flag		0 KO: Unsuccessful Operation 1 OK: Successful Operation that is, leaf pattern matches HashedKey
TSRx	Shared memory pool	Leaf contents are stored at TSEDPA address of the shared memory pool, specified by the tree search command operand.
LCBA0 / 1	Scalar registers	Leaf address is stored in LCBA0 / 1 based on LCBANr input value.
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 FM 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

Full Match (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 8-11*. 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 8-11. 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 with Dynamic_leaf_shape_en disabled	Single leaf associated with DTEntry; LCBA field contains pointer	Yes	Yes	01	LCBA	0
Pointer to leaf with Dynamic_leaf_shape_en enabled	Single leaf associated with DTEntry; LCBA field contains pointer NBT contains the LCBA shape and leaf shape from LuDefTable is ignored	Yes	Yes	01	LCBA	Bits (4..0) = LCBA Leaf Shape Bits (7..5) = Reserved

8.2.4.2 LPM Trees

Longest Prefix Match (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 picocode running on the GTH (and optionally the GDH) 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 8-12: LPM DTEntry and PSCBLine Formats* on page 297. 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.

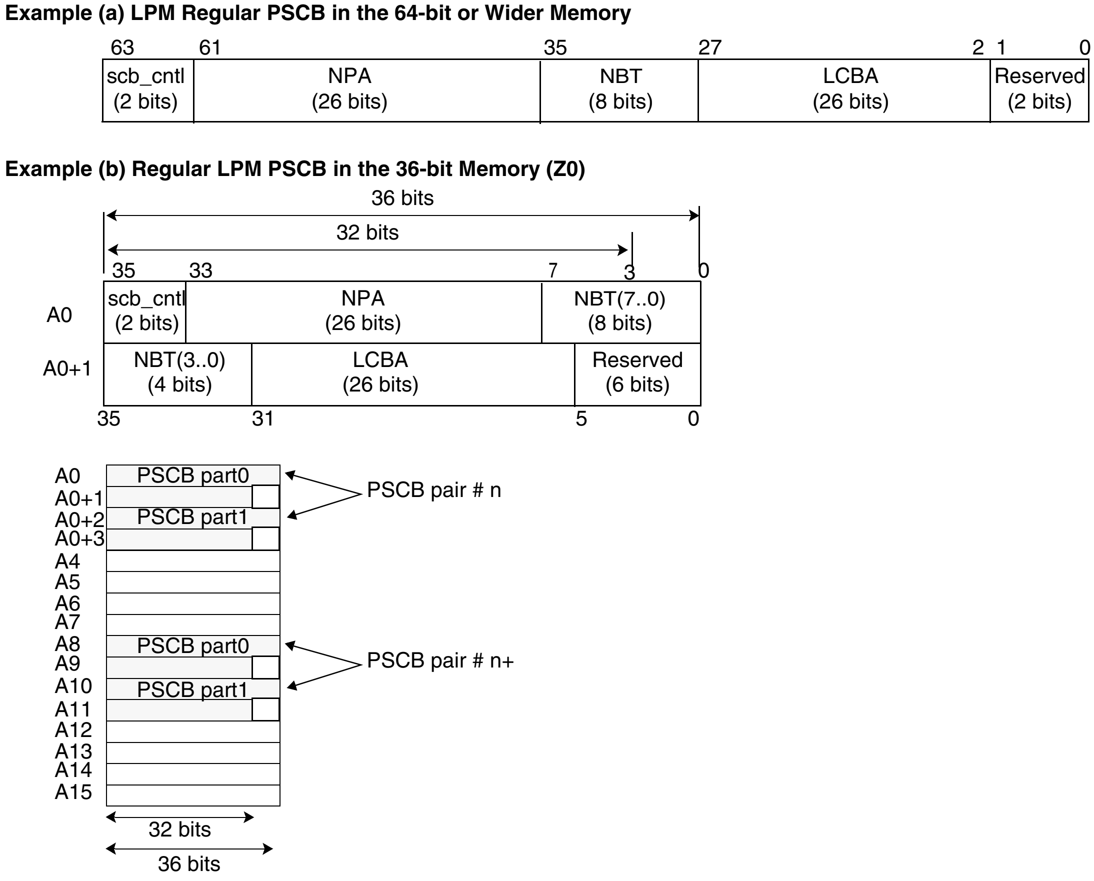
Table 8-12. LPM DTEntry and PSCBLine Formats

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 LPM PSCB Structure in Memory

Figure 8-3 on page 298 describes the layout of the LPM PSCB in the 64-bit or wider memory and the 36-bit memory (Z0).

Figure 8-3. Example of LPM PSCB Detail Layout in Memory



Note: A normal LPM PSCB pair is always a multiple of two addresses when stored in the 64-bit memory and a multiple of four addresses when stored in the 36-bit memory.

8.2.4.4 LPM Compact PSCB Support

LPM compact PSCB support is added to enhance memory usage for LPM PSCBs when stored in Z0 or H1 memories that have 36-bit wide data storage in each location. See *Figure 8-4* on page 300.

LPM PSCBs are in pairs and require a total of 64 bits to store a complete entry. When an LPM PSCB pair is stored in 64 bits or wider memory, it requires two consecutive locations and each PSCB part uses (1x1) shape in memory. When an LPM PSCB pair is stored in the 36-bit wide memory, it needs four sequential addresses to store complete PSCB pairs and each PSCB part uses (1x2) shape in the memory (that is, a complete PSCB entry will span to two addresses and a PSCB pair will span to four addresses. Also, a PSCB address must follow shape rules.

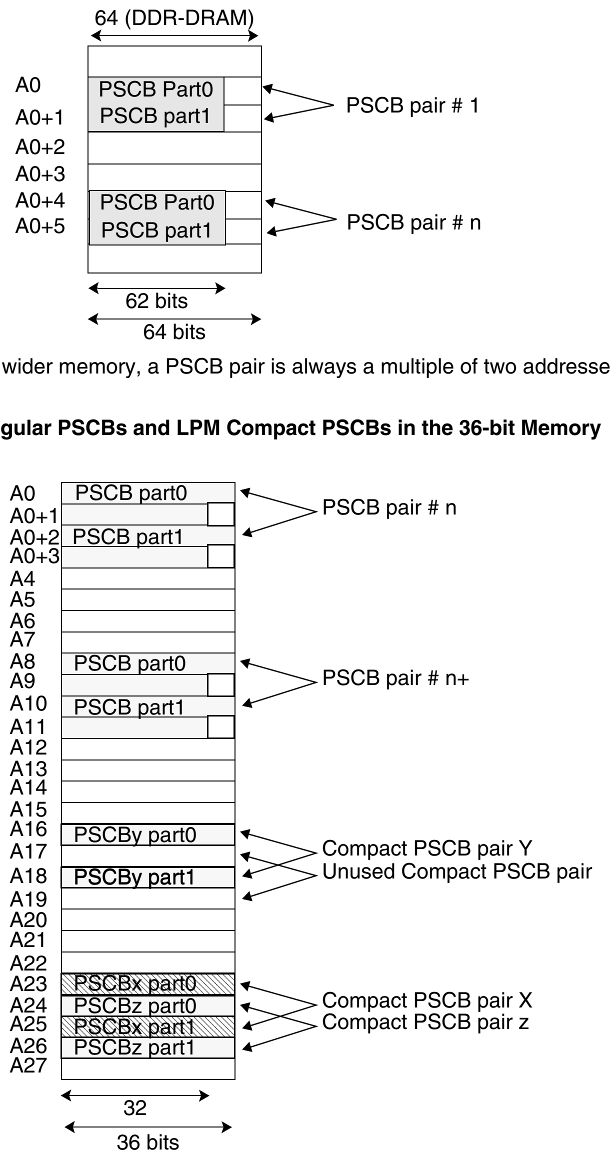
If LPM PSCBs are stored in a 36-bit wide memory, the format bits (NPA and NBT parts of the PSCB) will be stored in the first address (A0/A0+2) and LCBA is stored in the next address (A0+1/A0+3). If any intermediate LPM PSCBs do not have a valid LCBA (prefix), the LCBA part of the PSCB (A0+1) does not need to be stored. Using the LPM compact PSCB technique, one can make use of this (A0+1/A0+3) unused memory location. This requires software to manage two types of PSCB shapes:

1. Compact PSCBs, for intermediate PSCBs nodes with no prefix/LCBA
2. Full PSCB, for end node or intermediate PSCBs with prefix/LCBA

The compact PSCB uses address A+0 to store 0 part of PSCB (that is, format, NBT0, NPA0) and A+2 to store 1 part of PSCB (that is, format, NBT1, NPA1). Another independent PSCB pair can use A+1 for 0 part and A+3 for 1 part of PSCB.

8.2.4.5 LPM Trees with Multibit Compare

Figure 8-4. Example of LPM PSCB and Compact LPM PSCB Layout in Memory



Example (a) LPM Regular PSCBs in the 64-bit or Wider Memory

Note: In the 64-bit or wider memory, a PSCB pair is always a multiple of two addresses.

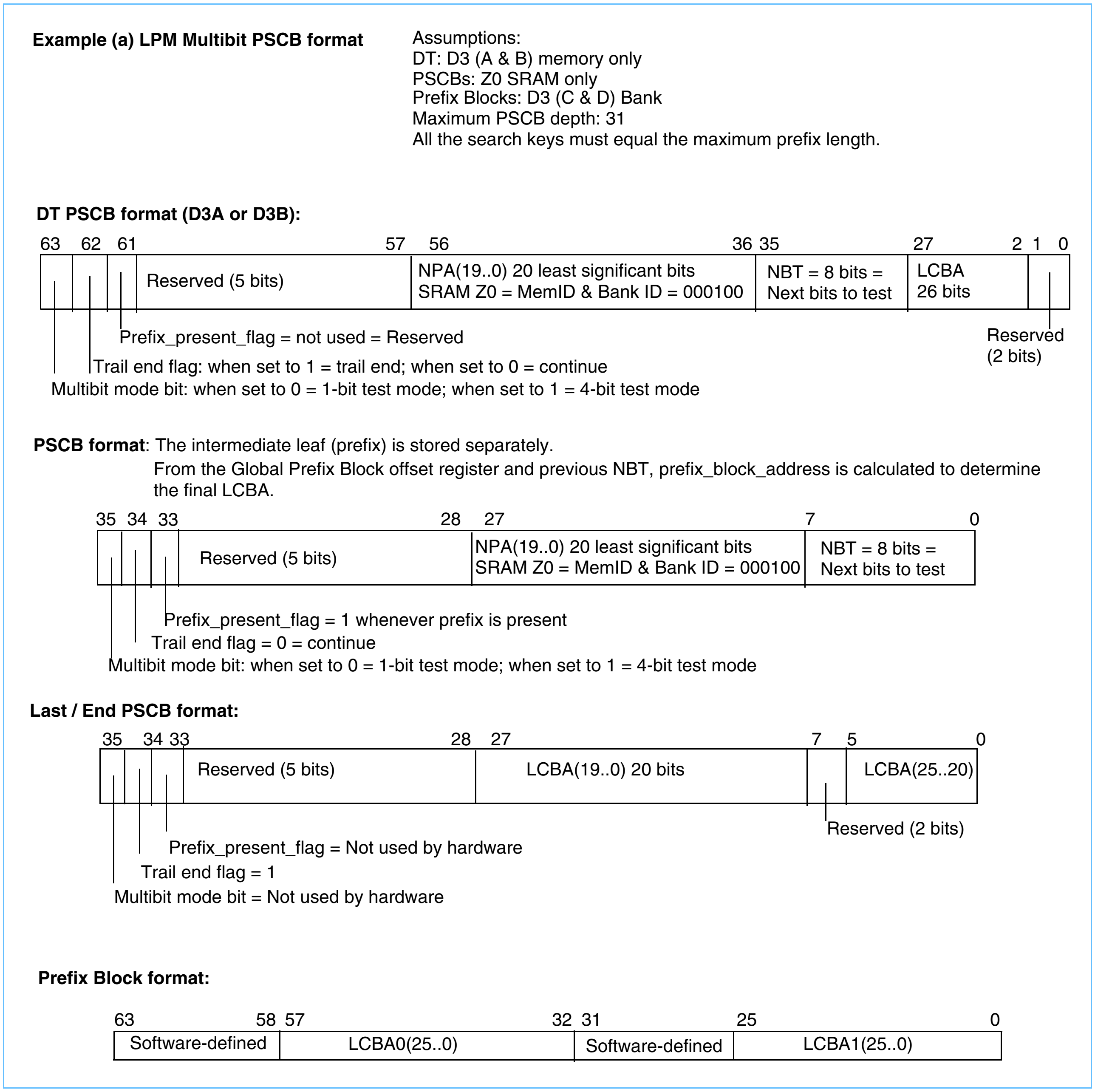
Example (b) LPM Regular PSCBs and LPM Compact PSCBs in the 36-bit Memory

Note: In the 36-bit memory, a normal LPM PSCB pair is always a multiple of four addresses, and an LPM compact PSCB pair is always allocated in a pair: An and An+2 *or* An and An+3.

As described earlier in *Section 8.2,* during a search, one bit is tested at every PSCB node. Except in the case of dense trees, it is possible to compare more than one consecutive bit (multibit mode) to reduce the number of PSCB nodes to be walked or compared. To enable the multibit compare mode, the Dynamic_multibit_ NBT_en bit defined in *Table 8-17: LUDefTable Entry Definitions* on page 307 must be set to ‘1’. *Figure 8-5* on page 302 describes various formats and their layout in memory. Also, the following assumptions apply to NP2G multibit LPM trees:

- LPM Multibit Direct Table can be stored in D3A or D3B only.
- LPM multibit PSCBs are stored in Z0 SRAM 36-bit wide memory only.
- LPM multibit Prefix Block (intermediate Leaf address) is stored in D3C or D3D (64-bit wide memory). For each entry in the PSCB block, LCBA (prefix address) is stored in a corresponding prefix block. Each prefix_block address contains two LCBAs. This address is derived using the Set Global Prefix Block Offset Register and NPA, and bits at the NBT offset.
  - Prefix_block_address_base = NPA(19..0) + Global_prefix_block_offset (19..0)
  - Prefix_Block_address = “10111” and Prefix_block_address_base(1) and (Prefix_block_address_base/4)
  - LCBA = prefix_block_data(57.32) when Prefix_block_address_base(0) = 0 and prefix_block_data(25..0) when Prefix_block_address_base(0) = 1
- Maximum PSCB depth: 31.
- All search keys should be equal to the longest prefix length.

Figure 8-5. LPM Multibit NBT PSCB Detail Layout



Example (a) LPM Multibit PSCB format

Assumptions:
DT: D3 (A & B) memory only
PSCBs: Z0 SRAM only
Prefix Blocks: D3 (C & D) Bank
Maximum PSCB depth: 31
All the search keys must equal the maximum prefix length.

DT PSCB format (D3A or D3B):

63	62	61	57	56 36	35	27 2	1 0
			Reserved (5 bits)	NPA(19..0) 20 least significant bits SRAM Z0 = MemID & Bank ID = 000100	NBT = 8 bits = Next bits to test	LCBA 26 bits	

Prefix_present_flag = not used = Reserved
Trail end flag: when set to 1 = trail end; when set to 0 = continue
Multibit mode bit: when set to 0 = 1-bit test mode; when set to 1 = 4-bit test mode
Reserved (2 bits)

PSCB format: The intermediate leaf (prefix) is stored separately.
From the Global Prefix Block offset register and previous NBT, prefix_block_address is calculated to determine the final LCBA.

35	34	33	28	27	7 0
			Reserved (5 bits)	NPA(19..0) 20 least significant bits SRAM Z0 = MemID & Bank ID = 000100	NBT = 8 bits = Next bits to test

Prefix_present_flag = 1 whenever prefix is present
Trail end flag = 0 = continue
Multibit mode bit: when set to 0 = 1-bit test mode; when set to 1 = 4-bit test mode

Last / End PSCB format:

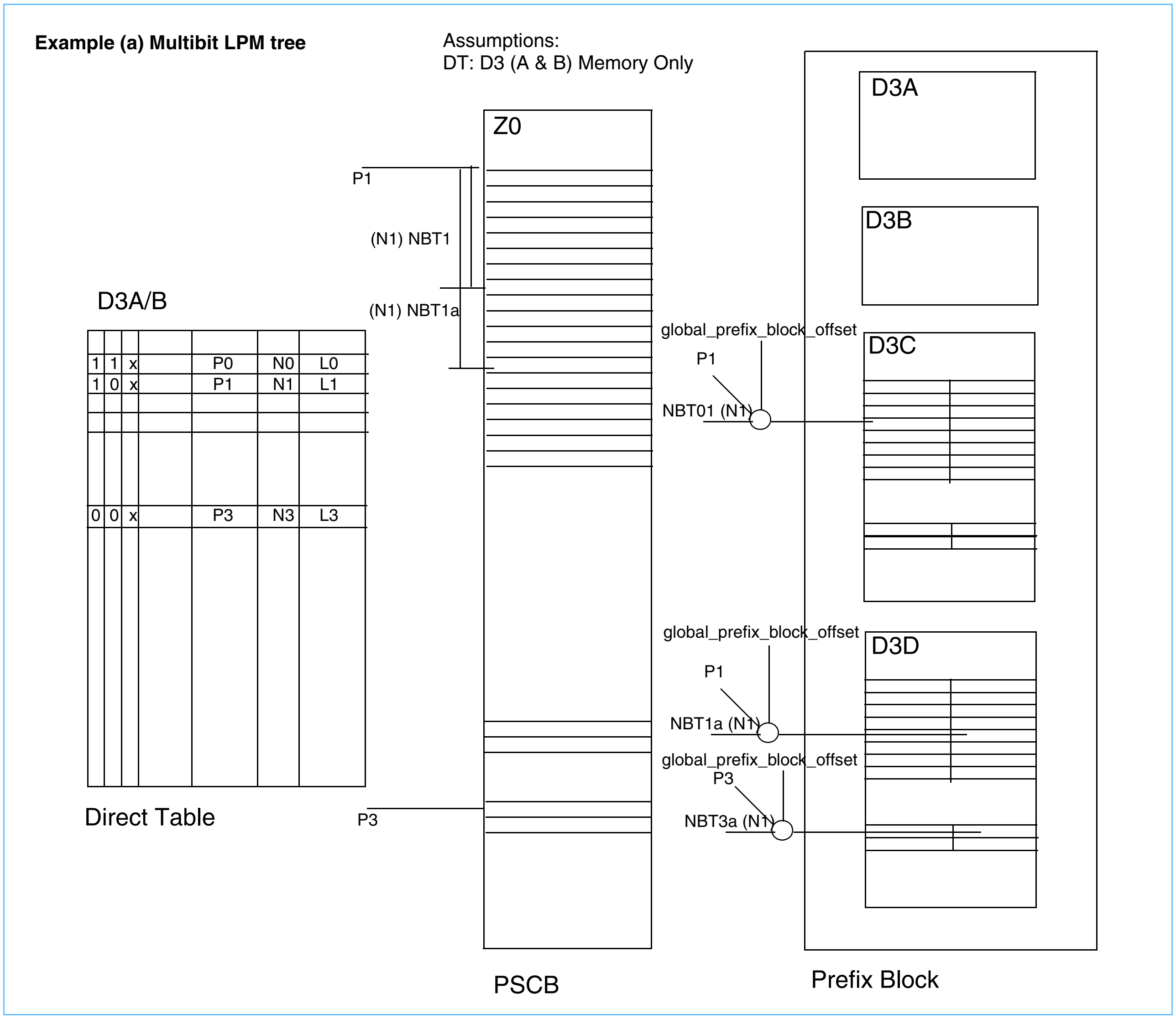
35	34	33	28	27	7	5 0
			Reserved (5 bits)	LCBA(19..0) 20 bits		LCBA(25..20)

Reserved (2 bits)
Prefix_present_flag = Not used by hardware
Trail end flag = 1
Multibit mode bit = Not used by hardware

Prefix Block format:

63 58	57 32	31 25	0
Software-defined	LCBA0(25..0)	Software-defined	LCBA1(25..0)

Figure 8-6. Example Multibit LPM Tree Layout



8.2.4.6 SMT Trees

Software-managed (SMT) trees provide a mechanism to create trees that follow a Control Point Function (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 8-17: LUDefTable Entry Definitions* on page 307. The following leaves in a leaf chain are shaped according to the five bits in the chaining pointer contained in the NLASMT leaf field (see *Table 8-13*). 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 8-13. NLASMT Field Format

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

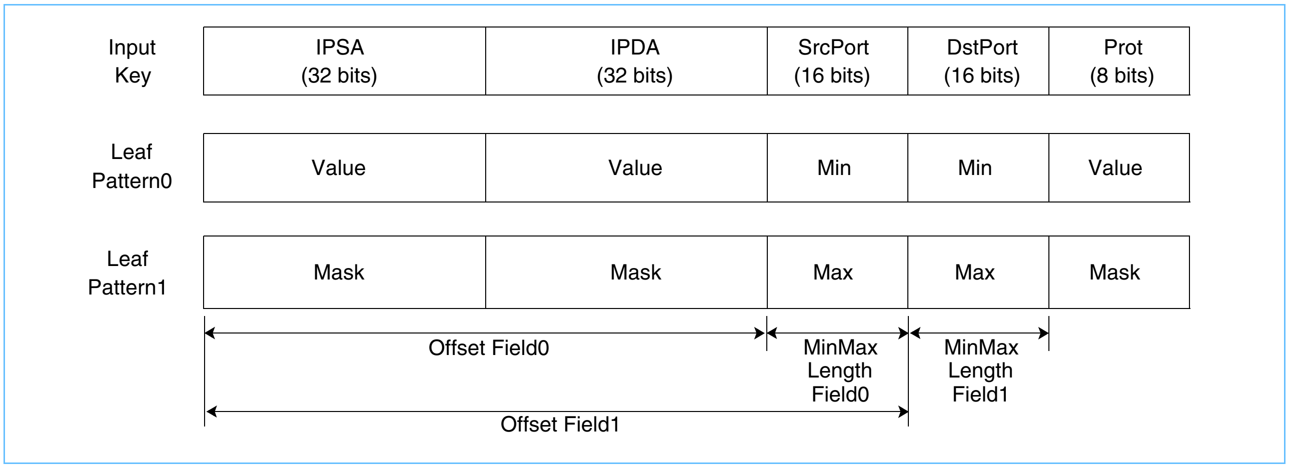
8.2.4.7 Compare-at-End Operation

The input key and the two reference patterns stored in each leaf can be logically divided into multiple fields (see *Figure 8-7* on page 304). 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. This operation uses CompTableDef for definition and type of compare indication.

Figure 8-7. Example Input Key and Leaf Pattern Fields



Logical field definitions are specified in the compare definition table CompDefTable. *Table 8-14* 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, 128, 144, 160, 174, or 192. The length of the field must be 8, 16, 24, or 32 bits.

Table 8-14. CompDefTable Entry Format

Field	Range	Bit Length	Value
Range 1 Offset (R1_Offset)	1	8	Starting bit position for first range compare. It is defined by compare_table(35 to 28) bits but only 4 upper bits are used to define the offset i.e (bits 35 to 32) and bits (31 to 28) are ignored. Therefore, this field must be in 16 bit boundary
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.This field is specified as the number of bits multiplied by 4. It is defined by compare_table(27 to 20) bits, but only 3 MSBs are used to define the length (i.e. bits 27,26 & 25) and bits 24 to 20 are ignored.
Range 2 Offset (R2_Offset)	2	8	Starting bit position for second range compare. It is defined by compare_table(19 to 12) bits but only 4 upper bits are used to define the offset; i.e bits 19 to 16 and bits 15 to 12 are ignored. Therefore, this field must be in 16-bit bound.
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. It is defined by compare_table(11 to 4) bits but only 3 MSBs are used to define the length (i.e. bits 11 to 9) and bits (8 to 4) are ignored.
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. It is defined by compare_table(3) bit.
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 It is defined by compare_table(2) bit.

In the input key shown in *Figure 8-7: Example Input Key and Leaf Pattern Fields* on page 304, 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 8-7: SMT Tree Fixed Leaf Formats* on page 292). 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.

8.2.4.8 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 8-15: LUDefTable Rope Parameters* on page 306.

Figure 8-8. Rope Structure

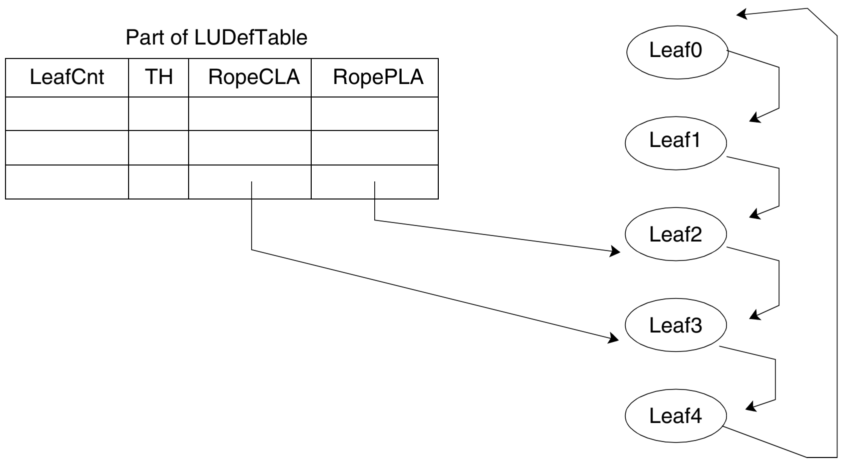


Table 8-15. LUDefTable Rope Parameters

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 333) relate to the leaf addressed by RopeCLA.
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:

1. 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.
2. When the rope is being walked and the DeletePending bit is set, the TSE deletes the leaf automatically.

8.2.4.9 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 that matches the leaf pattern is found during a tree search, 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 8-16. 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 8-17. LUDefTable Entry Definitions (Page 1 of 4)

Field	Bit Length	Description
Cascade_entry/ Cache_entry	1	0 Normal tree entry (not a cache entry) 1 A Cascade/cache entry. When a search returns with KO, and Cache_entry = 1, TS instructions will restart using the next entry in the LUDefTable (that is, LUDefIndex + 1). The last LUDefIndex in the search chain entry must have this bit set to '0' (disabled) when more than 1 table is cascaded. Cascading of more than 1 table is allowed.
Search_Type	2	Search Type value. This field indicates the type of search to be performed. 00 Full Match (FM). 01 Longest Prefix Match (LPM). 10 Software Managed Tree (SMT). 11 Reserved.
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 Reserved 1000 8-bit Hash 1001 12-bit Hash 1010 16-bit Hash **Note:** When using Hash_Type 1000, 1001, and 1010, Hashedkey and HashedKey_length will be extended by 8, 12, or 16 bits, respectively.
Color_En	1	Color Enable control value. This field controls whether the Color value is used to form the hashed key value. 0 Color register not used. 1 Color register is used

Table 8-17. LUDefTable Entry Definitions (Page 2 of 4)

Field	Bit Length	Description
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. P1P2_Max_size mapping: 00000 0 byte (no pattern) 00001 2 byte 00010 4 bytes 00011 6 bytes 00100 8 bytes 00101 10 bytes 00110 12 bytes 00111 14 bytes 01000 16 bytes 01001 18 bytes 01010 20 bytes 01011 22 bytes 01100 24 bytes Others 0 byte (Reserved) **Notes:** For FM: Unused bits between p1p2_max_size - hkey_length must be initialized to zero. For LPM: Unused bits between (p1p2_max_size - prefix_length) do not have to be initialized and can be used by user data since these bits do not take part in the comparison. For SMT: Unused bit between (2 * p1p2_max_size - 2 * Hashedkey_length) must be initialized to zero.
NLARope_En	1	NLA Rope Enable control value. This field indicates whether or not the leaf contains a Next Leaf Address Rope (NLA Rope) field. 0 Leaf does not contain NLARope field 1 Leaf contains NLARope field (enables aging)
PSCB_FQ_Index	6	Pattern Search Control Block Free Queue Index value. This is the index of the PSCB free list.
Reserved	1	Reserved.
Dynamic_multibit_NBT_en	1	When set to ‘1’, multibit NBT PSCB format is used during the search, enabling 1- or 4-bit comparison at each PSCB node. When set to ‘0’, single-bit PSCB format (legacy) is used and only one bit will be tested at each PSCB node. This mode bit is available in *NP2G*
DT_Base_Addr	26	Direct Table Base Address value. **Note:** This field stores flat memory base_address for the MRD and MWR commands with Shape_cntl = 01 (flat memory read/write). This base address and Leaf_Height are used to compute the final address.

Table 8-17. LUDefTable Entry Definitions (Page 3 of 4)

Field	Bit Length	Description
DT_Size	4	Direct Table Size value - defines the number of DT entries within a memory bank. 0001 4 entries (2 bits) 0010 16 entries (4 bits) 0011 64 entries (6 bits) 0100 256 entries (8 bits) 0101 1 K entries (10 bits) 0110 4 K entries (12 bits) 0111 16 K entries (14 bits) 1000 32 K entries (15 bits) 1001 64 K entries (16 bits) 1010 128 K entries (17 bits) 1011 256 K entries (18 bits) 1100 512 K entries (19 bits) 1101 1 M (20 bits) Others Reserved
Leaf_FQ_Index	6	Defines index of leaf free list
Leaf_Width	2	Leaf width
Leaf_Height	3	Leaf height
Dynamic_leaf_shape_en	1	This option is valid for the TS_SMT and TS_FM commands. When this mode is enabled, each primary leaf can have independent shape (size). When this option is set to ‘0’, for the FM and SMT search commands, the leaf shape is defined as per the LUDef leaf_width and leaf_height parameters. Therefore, each primary leaf has the same shape. When this option is set to ‘1’, the five least significant bits of the NBT field are used to define the shape for that leaf. NBT bits 2:0 define the leaf_height parameter and bits 4:3 define the leaf_width. Also, the leaf_height and leaf_width parameters of the LUDef table are ignored. This mode bit is available in *NP2G*.
Leaf_read_Disable	1	This option is valid for the TS_LPM command. When this option is set to ‘1’, the LPM search algorithm (at the end of a successful search) will not write the leaf contents to the Shared Memory Pool, and only the leaf address is returned. A separate picocode memory read instruction is required to read the leaf content. When this option is set to ‘0’, the LPM search algorithm (at the end of the search) writes the leaf contents back to the Shared Memory Pool and returns the leaf address. This mode bit is available in *NP2G*.
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.
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.

Table 8-17. LUDefTable Entry Definitions (Page 4 of 4)

Field	Bit Length	Description
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 or equal to this assigned threshold, then a class 0 interrupt is generated (bit 9 of the class 0 interrupt vector). Checking of the threshold is performed only for the TLIR rope command. 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) = 2**(LeafTh(9:5))or 2**(LeafTh(4:0))

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 8-18*. 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 335 and *Section 8.2.8.4 Tree Search Dequeue Free List (TSDQFL)* on page 336.

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 8-18. 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 initialized to 0. The threshold is determined as 2**Threshold. When the QCnt is less than or equal to the threshold, a Class 0 interrupt (bit 8) is generated.

8.2.6 TSE Registers and Register Map

Table 8-19. TSE Scalar Registers for GTH Only (Page 1 of 2)

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 293 and Section *8.2.2 Input Key and Color Register for SMT Trees* on page 293.
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 Cascade/Cache* on page 295.
NrPSCBs	R	08	8	See *Section 8.2.3.3 Cascade/Cache* on page 295.
Global_prefix_block_ offset	R/W	09	26	See *Section 8.2.4.5 LPM Trees with Multibit Compare* on page 300.
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

Table 8-19. TSE Scalar Registers for GTH Only (Page 2 of 2)

Name	Read/ Write	Hex Address	Bit Length	Description
LUDefCopy	R	10-12	96	Contains LU_Def_Table index in use and a copy of the following LU_Def_Table fields: Field Bits Definition reserved 95:88 x‘00’ reserved LU_Def_Index 87:80 Index of the location from where the content was read. For remaining field definitions, see *Table 8-17: LUDefTable Entry Definitions* on page 307, with exceptions noted. LU_Def_Invalid 79 LU_Def_State 78:77 LPM_leaf_read_disable 76 Dynamic_leaf_shape_en 75 Leaf_Height 74:72 Leaf_Width 71:70 Leaf_FQ_Index 69:64 DT_Size 61:58 DT_Base_Addr 57:32 Reserved 31:24 Dynamic_multibit_NBT_en 23 Reserved 22 PSCB_FQ_Index 21:16 Reserved 15:14 reserved NLA_Rope_Ena 13 P1P2_Max_Size 12:8 Color_Ena 7 Hash_Type 6:3 Search_Type 2:1 Cascade/Cache Entry 0
LUDefCopy 95_64	R	10	32	Bits 95:64 of the LUDefCopy register
LUDefCopy 63_32	R	11	32	Bits 63..32 of LUDefCopy
LUDefCopy 31_0	R	12	32	Bits 31..0 of LUDefCopy
SetPatbit_GDH_4bit	R/W	1C	4	Contains 4 bits from the pattern stored in TSRx. Set by the SetPatBit_GDH command.
SetPatbit_GDH	R/W	1B	1	Contains one bit from the pattern stored in TSRx. Set by SetPatBit_GDH command.
DistPosReg_GDH	R	1A	8	Contains result of DISTPOS command from DistPos_GDH
LUDefCopy_GDH	R	19	26	Contains LUDefTable index in use and a copy of the following LUDefTable fields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24, 23) Reserved LUDefCopy_GDH(22, 21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0)
LUDefCopy_GDH 31_0	R	19	26	Bit 31..0 of LUDefCopy_GDH

Table 8-20. 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 a given thread's starting address in the Shared Memory Pool.					

Table 8-21. 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 8-22. 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 Cascade/Cache* on page 295
NrPSCBs	R	08	8	See *Section 8.2.3.3 Cascade/Cache* on page 295.
SetPatbit_GDH	R	1B	1	Contains one bit from the pattern stored in TSRx. Set by SetPatBit_GDH command.
Global_prefix_block_ offset	R/W	09	26	See *Section 8.2.4.5 LPM Trees with Multibit Compare* on page 300.

Table 8-22. TSE Scalar Registers for GDH and GTH

Name	Read/ Write	Hex Address	Bit Length	Description
SetPatbit_GDH_4bit	R	1C	4	Contains four bits from the pattern stored in TSRx. Set by the SetPatBit_GDH command.
DistPosReg_GDH	R	1A	8	Contains result of DISTPOS command from DistPos_GDH
LUDefCopy_GDH	R	19	26	Contains LUDefTable index in use and a copy of the following LUDefTable fields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24,23) Reserved LUDefCopy_GDH(22,21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0)

Table 8-23. 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 8-24. TSE GTH Indirect Registers (Page 1 of 2)

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

Table 8-24. TSE GTH Indirect Registers (Page 2 of 2)

Indirect Register	Bit Length	Description	Notes
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 311. 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 8-25. 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

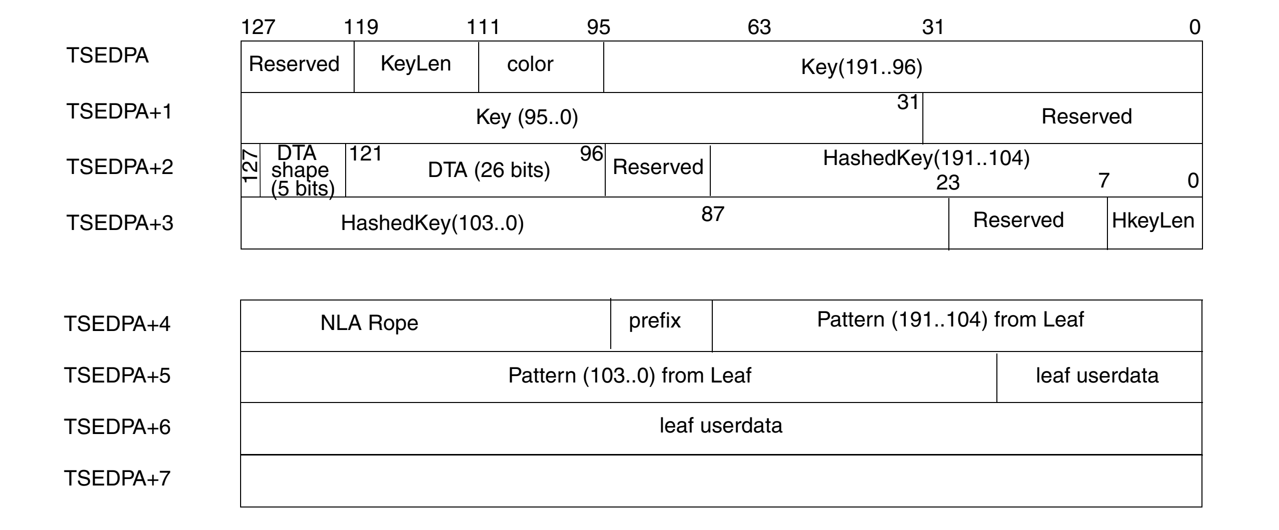
The Tree Search Engine (TSE) provides facilities for conducting and managing tree searches. Two TSE coprocessor interfaces are provided for each thread: TSE0 and TSE1. Therefore, each thread can issue instructions to each of these TSE0 and TSE1 resources.

General TSE instructions can be executed by both TSE0 and TS1 in any thread.

Table 8-26. General TSE Instructions

Opcode	Command	Detail Section
0	Null	
1	TS_FM	*8.2.7.1 FM Tree Search (TS_FM)* on page 317
2	TS_LPM	*8.2.7.2 LPM Tree Search (TS_LPM)* on page 318
3	TS_SMT	*8.2.7.3 SMT Tree Search (TS_SMT)* on page 320
4	MRD	*8.2.7.4 Memory Read (MRD)* on page 321
5	MWR	*8.2.7.5 Memory Write (MWR)* on page 322
6	HK	*8.2.7.6 Hash Key (HK)* on page 323
7	RDLUDEF	*8.2.7.7 Read LUDefTable (RDLUDEF)* on page 324
8	COMPEND	*8.2.7.8 Compare-at-End (COMPEND)* on page 326
9	DistPos_GDH	*8.2.7.9 Distinguish Position for Fast Table Update (DISTPOS_GDH)* on page 327
10	RDPSCB_GDH	*8.2.7.10 Read PSCB for Fast Table Update (RDPSCB_GDH)* on page 328
11	WRPSCB_GDH	*8.2.7.11 Write PSCB for Fast Table Update (WRPSCB_GDH)* on page 330
12	SetPatBit_GDH	*8.2.7.12 SetPatBit_GDH* on page 331
13 to 15	Reserved	
Note: Commands can be executed by all GxHs with threads.		

Figure 8-9. General Layout of TSE Fields in Shared Memory Pool



8.2.7.1 FM Tree Search (TS_FM)

Table 8-27. 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 *7.2.4 Shared Memory Pool* on page 172) 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		Pattern to be searched, located in Shared Memory Pool. The key must be initialized only in the byte boundary of the key length.
KeyLength	8	Shared Memory Pool		Length of pattern minus 1 in key. Must be initialized before search. Located in shared memory pool.
Color	16	Shared Memory Pool		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'.

Table 8-28. FM Tree Search Results (TSR) Output (Page 1 of 2)

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 4 QW past the starting QW indicated by the input TSEDPA parameter. That is, the Shared Memory Pool QW location = TSEDPA*4 + 4
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 Cascade/Cache* on page 295.
NrPSCBs	8	Register	See Section *8.2.3.3 Cascade/Cache* on page 295.

Table 8-28. FM Tree Search Results (TSR) Output (Page 2 of 2)

Result	Bit Length	Source	Description
LUDefCopy_GDH	26	Register	Contains LUDefTable index in use and a copy of the following LUDefTablefields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24,23) Reserved LUDefCopy_GDH(22,21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0) Will be stored in scalar register and its address in Register Address Map is ‘019’.
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 8-29. LPM Tree Search Input Operands (Page 1 of 2)

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)	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 *7.2.4 Shared Memory Pool* on page 172) 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		Pattern to be searched, located in Shared Memory Pool. The key must be initialized only in the byte boundary of the key length.
KeyLength	8	Shared Memory Pool		Length of pattern minus 1 in key. Must be initialized before search. Located in shared memory pool.
Color	16	Shared Memory Pool		Used only when enabled in LUDefTable. Must be initialized before search. Located in Shared Memory Pool.
global_prefix_block_offset	20	Register		Contains global prefix offset used by Multibit LPM trees. Required when Dynamic_multibit_NBT_en is enabled. Will be stored in scalar register, and its address in the Register Address Map is X‘009’.

Table 8-29. LPM Tree Search Input Operands (Page 2 of 2)

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
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. **Note:** UseLUDefCopyReg = ‘1’ is not supported for LUDefEntry with Cache_enable.
LUDefCopy	96	Register		Input only when UseLUDefCopyReg is ‘1’. Set to contents of LUDefTable at entry given by LUDefIndex.

Table 8-30. LPM Tree Search Results 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 4 QW past the starting QW indicated by the input TSEDPA parameter. That is, Shared Memory Pool QW location = TSEDPA*4 + 4
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 Cascade/Cache* on page 295
NrPSCBs	8	Register	See Section *8.2.3.3 Cascade/Cache* on page 295
LUDefCopy_GDH	26	Register	Contains LUDefTable index in use and a copy of the following LUDefTablefields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24,23) Reserved LUDefCopy_GDH(22,21) LP1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0) Will be stored in scalar register, and its address in the Register Address Map is ‘019’.
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 8-31. 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)	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 *7.2.4 Shared Memory Pool* on page 172) 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		Pattern to be searched, located in shared memory pool. The key must be initialized only in the byte boundary of the key length.
KeyLength	8	Shared Memory Pool		Length of pattern minus 1 in key. Must be initialized before search. Located in shared memory pool.
Color	16	Shared Memory Pool		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. **Note:** UseLUDefCopyReg = ‘1’ is not supported for LUDefEntry with Cache Enable.
LUDefCopy	96	Register		Input only when UseLUDefCopyReg is ‘1’. Set to contents of LUDefTable at entry given by LUDefIndex.

Table 8-32. SMT Tree Search Results Output (Page 1 of 2)

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 4 QW past the starting QW indicated by the input TSEDPA parameter. That is, Shared Memory Pool QW location = TSEDPA*4 + 4
LCBA0 / 1	26	Register	When OK is ‘1’, leaf address is stored in LCBA0 / 1
CacheFlags	3	Register	See Section *8.2.3.3 Cascade/Cache* on page 295
NrPSCBs	8	Register	See Section *8.2.3.3 Cascade/Cache* on page 295

Table 8-32. SMT Tree Search Results Output (Page 2 of 2)

Result	Bit Length	Source	Description
LUDefCopy_GDH	26	Register	Contains LUDefTable index in use and a copy of the following LUDefTablefields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24,23) Reserved LUDefCopy_GDH(22,21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0) Will be stored in scalar register, and its address in Register Address Map is '019'.
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. When flat memory mode (shape_cntl = 01) is selected, the 20 least significant bits in LCBA0/1 contain the offset that is used with leaf shape and DT_Base_Addr from the LUDefTable to compute the final address to be read. In flat memory mode, the following rules are applied:

- The 20 least significant bits in LCBA0/1 can be used as an offset; the remaining bits are ignored.
- Valid leaf shape heights are 1, 2, and 4; no other heights are supported.
- During final address computation, address spanning is not allowed.

Flat memory mode is valid for NP2G.

Table 8-33. Memory Read Input Operands (Page 1 of 2)

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
ShapeCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct shape. 01 Flat mode. Leaf_shape_height and DT_base_addr from LuDefTable are used with the LCBA0/1 field to compute the address to be read. Leaf shape from LuDefTable is used as a shape. When this mode is selected, the final address to be read = DT_base_addr + (LCBA0/1 * Leaf_shape_height). 10 PSCB shape is based on address and tree type from LudefTable. That is, LPM tree type and ZBT or H1 address shape will be set to (1x2), and for any other case, shape will be set to (1x1). 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 '01', '10' or '11'.
Width	2	Imm16(9..8)	GPR(4..3)	Width of object to be read. Valid only when ShapeCtrl is '00'.

Table 8-33. Memory Read Input Operands (Page 2 of 2)

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
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)	TSE Thread Shared Memory Pool Address determines the destination location for the data to be read. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) 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 when ShapeCtrl is set to ‘00’, ‘10’, or ‘11’. This operand contains the offset used for address computation when ShapeCtrl is set to ‘01’.

Table 8-34. 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. When flat memory mode (shape_cntl = 01) is selected, the 20 least significant bits in LCBA0/1 contain the offset that is used with leaf shape and DT_Base_Addr from the LUDefTable to compute the final address to be read. In flat memory mode, the following rules are applied:

- The 20 least significant bits in LCBA0/1 can be used as an offset; the remaining bits are ignored.
- Valid leaf shape heights are 1, 2, and 4; no other heights are supported.
- During final address computation, address spanning is not allowed.

Flat memory mode is valid for NP2G.

Table 8-35. Memory Write Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
ShapeCtrl	2	Imm16(14..13)	GPR(9..8)	00 Direct Shape 01 Flat mode. Leaf_shape_height and DT_base_addr from LuDefTable are used with LCBA0/1 field to compute the address to be written. Leaf shape from LuDefTable is used as a shape. When this mode is selected, final address to be written = DT_base_addr + (LCBA0/1 * Leaf_shape_height). 10 PSCB shape is based on address and tree type from LudefTable. That is, LPM tree type and ZBT or H1 address shape will be set to (1x2), and for any other case, shape will be set to (1x1). 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 ‘01’, ‘10’ or ‘11’.
Width	2	Imm16(9..8)	GPR(4..3)	Width of object to be written. Valid only when ShapeCtrl is ‘00’.
Height	3	Imm16(7..5)	GPR(2..0)	Height of object to be written. Valid only when ShapeCtrl is ‘00’.
TSEDPA	4	Imm16(4..1)	Imm12(4..1)	TSE Thread Shared Memory Pool Address determines the source location for the data to be written. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) 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 read when ShapeCtrl is set to ‘00’, ‘10’, or ‘11’. This operand contains the offset used for address computation when ShapeCtrl is set to ‘01’.
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 8-36. 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)	TSE Thread Shared Memory Pool Address - stores location of Key, Key-Length, and Color and determines Hash Key destination. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) 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

Table 8-36. Hash Key Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
Key	192	Shared Memory Pool		Pattern to be searched, located in Shared Memory Pool. The key must be initialized only in the byte boundary of the key length.
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 8-37. 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 293 and Section *8.2.2 Input Key and Color Register for SMT Trees* on page 293. Hash function is defined in the LUDefTable. Stored in the Shared Memory Pool QW location = TSEDPA*4 + 2 & 3.
HashedKeyLen	8	Shared Memory Pool	Hashed key length contains the length of pattern minus 1 in HashedKeyReg. Stored in the shared memory pool QW location = TSEDPA*4 + 3 bit (7:0).
DTA	26	Shared Memory Pool	DTEntry Address. Stored in shared memory pool QW location = TSEDPA*4 + 2, bits (121:96). **Note:** Not valid when Direct_HashType_En is set to value ‘1’.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation
LUDefCopy_GDH	26	Register	Contains LUDefTable index in use and a copy of the following LUDefTablefields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24,23) Reserved LUDefCopy_GDH(22,21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0) Will be stored in scalar register, and its address in Register Address Map is ‘019’.
The 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 field in the shared memory pool.The TSE can read LUDefTable while picocode builds a key because RDLUDEF is executed asynchronously.

Figure 8-10. General Layout of TSE RDLUDEF in Shared Memory Pool

<table>
<tr><td></td><td>127</td><td>95 120</td><td>119 112</td><td>111 32</td><td>31 0</td></tr>
<tr><td>TSEDPA</td><td colspan="2">Reserved</td><td>LuDefIndex</td><td>LUDefTable Entry (111..32)</td><td>Reserved</td></tr>
<tr><td>TSEDPA+1</td><td>Reserved</td><td colspan="4">LUDefTable_Rope Entry (95 ..0) (GTH only)</td></tr>
</table>

Table 8-38. 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)	TSE Thread Shared Memory Pool Address determines the destination location for the LUDefTable entry data to be read. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary.

Table 8-39. RDLUDEF Output Results

Result	Bit Length	Source	Description
LUDefCopy	88	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. Various fields of LUDefEntry are at the same bit position as in the LUDefTable memory. (111 to 32) bits represent the LUDefEntry contents and bits (119 to 112) represents the corresponding LUDefIndex. They are left-justified. **Note:** TSEDPA address and TSEDPA + 1 will be overwritten during this command. Contents of LUDefCopy will be returned at TSEDPA and contents of LUDefCopy_Rope will be returned in the next address. LUDefCopy_Rope content will be valid for the GTH thread.
LUDefCopy_GDH	26	Register	Contains LUDefTable index in use and a copy of the following LUDefTablefields: Dynamic_leaf_shape_en LUDefCopy_GDH(25) Reserved LUDefCopy_GDH(24, 23) Reserved LUDefCopy_GDH(22, 21) P1P2_max_size LUDefCopy_GDH(20 to 16) DT_size LUDefCopy_GDH(15 to 12) NLARope_en LUDefCopy_GDH(11) color_en LUDefCopy_GDH(10) Tree_Type LUDefCopy_GDH(9,8) LuDefTable_index LUDefCopy_GDH(7 to 0) Will be stored in scalar register and its address in Register Address Map is ‘019’.
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 information, and the second could contain leaves with quality of service information. COMPEND should be used only after a tree search command. The COMPEND command uses Key instead of HashedKey during operation, since it is assumed that hashing is not needed for SMT. Hence, the Key field is read by the command, not the HashedKey field. For NP2G, it is not necessary to initialize the unused key bits to zero.

Table 8-40. 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)	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 *7.2.4 Shared Memory Pool* on page 172) 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.
Key	192	Shared Memory Pool		Input of previous tree search command located in the Shared Memory Pool at an offset of 0 and 1QW from the QW indicated by the TSEDPA.
KeyLen	8	Shared Memory Pool		Input of previous tree search command located in the Shared Memory Pool at the QW indicated by the TSEDPA

Table 8-41. 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.

8.2.7.9 Distinguish Position for Fast Table Update (DISTPOS_GDH)

DISTPOS_GDH performs a pattern compare between the pattern stored in HashedKey of the shared memory pool and a pattern from the leaf specified in TSRx. The result is stored in the DistPosReg_GDH register. The OK flag is set when a full match has been detected.

Figure 8-11. Shared Memory Pool with DISTPOS_GDH Command Subfields

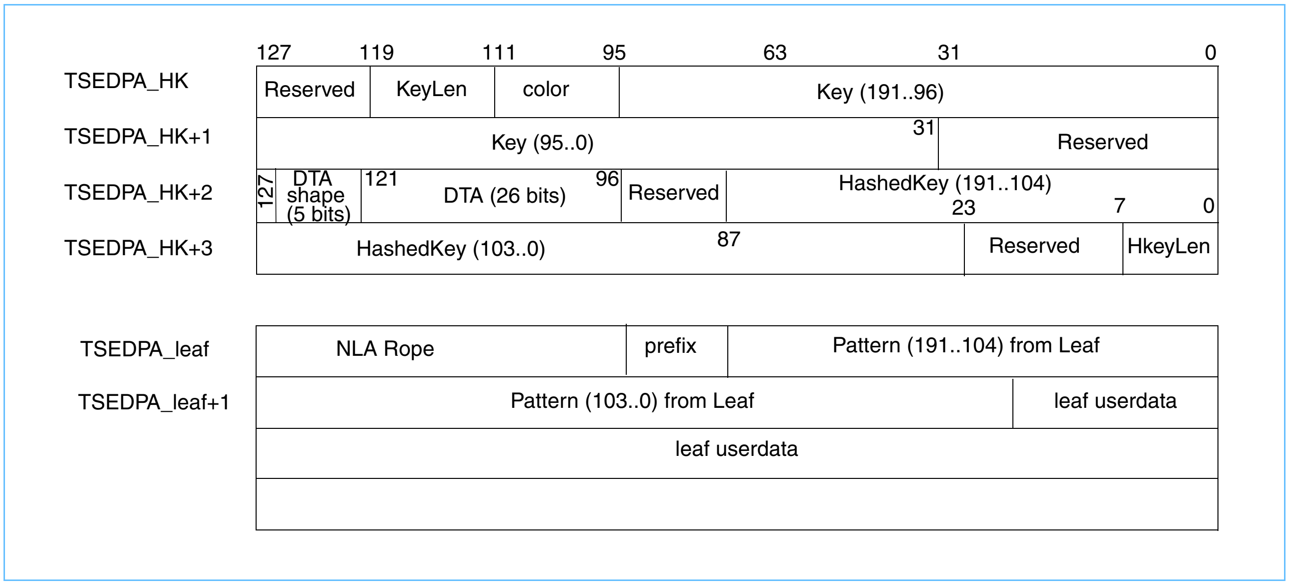


Table 8-42. DISTPOS_GDH Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
TSEDPA_leaf	4	Imm16(4..1)	Imm12(4..1)	Indicates the leaf storage location in the Thread Shared Memory Pool. The TSEDPA_leaf is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary. Use only values of x '8', x 'A', x 'C', and x 'E'.
TSEDPA_HK	4	Imm16(8..5)	Imm12(8..5)	Indicates the HashedKey storage location in the Thread Shared Memory Pool. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see 7.2.10 Shared Memory Pool) and is constrained to be on a four-QW boundary. Use only values of x '8', x 'A', x 'C', and x 'E'. Hashedkey is assumed to be stored at TSEDPA +2 location in shared memory pool.
HashedKey	192	Shared Memory Pool		Contains the hashed pattern and it will be stored at TSEDPA_HK +2 (7..0) quad word location. The structure of the Hashed Key in shared memory pool is shown in *Figure 8-11*.
HashedKeyLen	8	Shared Memory Pool		Contains length of hashed pattern minus 1. Stored at TSEDPA_HK +2 quad word location. Structure of Hashed Key in shared memory pool is shown in *Figure 8-11*.
TSRx	512	Shared Memory Pool		Contains second pattern with pattern length (for LPM & FM only). TSRx is mapped into the shared memory pool, starting at an offset of 4 QW past the QW indicated by the input TSEDPA_leaf parameter for a length of 4 QW.

Table 8-43. DISTPOS_GDH Output Results

Result	Bit Length	Source	Description
OK/KO	1	Flag	0 Pattern does not match 1 Pattern in HashedKey matches pattern in TSRx
DistPosReg_GDH	8	Scalar register	Smallest bit position where pattern in HashedKey differs from pattern in TSRx. Will be stored in scalar register and its address in the Register Address Map is ‘01A’.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

DISTPOS_GDH applies to LPM and FM only. It needs P1P2_Max_Size value as a part of the computation. The value is assumed to be valid prior to this command. This value is stored as a scalar register, and is independent for each thread.

8.2.7.10 Read PSCB for Fast Table Update (RDPSCB_GDH)

RDPSCB_GDH reads a PSCB from the control store and stores it in one of the PSCBx locations in the shared memory pool. For LPM, the entire PSCB is read from memory at the address given by PSCB.Addr and stored in PSCB. For FM, the entire PSCB is read from memory and converted into a LPM PSCB format and either NPA0/1 or LCBA0/1 will be set to all zero, since only one of the fields will be valid. This command is not supported for SMT.

Structure for PSCB register shared memory pool is as follows:

Figure 8-12. Shared Memory Pool with PSCB Subfields

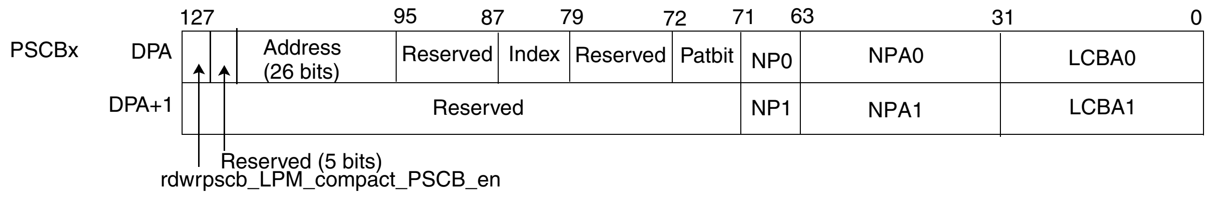


Table 8-44. RDPSCB_GDH Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
DPA_PSCB	6	Imm16(5..0)	Imm12(5..0)	The DPA_PSCB is a Full address of the shared memory pool (see *Section 7.2.4 Shared Memory Pool* on page 172). This indicates working address of the PSCB register. Full PSCB will occupy 2 quadwords from shared memory pool. This address must start at an even address.
RdWrPSCB_Cntl	2	Imm16(7..6)	Imm12(7..6)	00 Result is based on PatBit value in the shared memory pool as shown in above diagram. 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
Reserved	1	Imm16(8)	Imm12(8)	Reserved.
PSCB.Addr	26	Shared Memory Pool (121..96)		Address to be read in control store of the PSCB. This is stored in shared memory pool.
PSCB.rdwrps_LPM_compact_PSCB_en	1	Shared Memory Pool (127)		This optional argument for LPM reduces memory read bandwidth. When not used, it must be set to ‘0’. This option enables use of the LPM compact PSCB. It indicates the LPM DT/PSCB entry has no valid leaf address and is stored in memory that is 36 bits wide or less than 64 bits wide (that is, Z0 and H1 SRAM memories). It is not used by FM or SMT. 0 For FM PSCB/DT and when LPM DT/PSCB is stored in 64-bit or wider memory for width = 1 and height = 1 (DDR DRAM memory). 1 When LPM DT/PSCB is stored in memory that is narrower than 64 bits for width = 1 and height = 1 (36-bit Z0 and H1 SRAM memories). When this argument is set to ‘1’, PSCB.Addr+0 and PSCB.Addr+2 are treated as one PSCB pair. PSCB.Addr+1 and PSCB.Addr+3 are treated as separate PSCB pair. Addresses are interleaved for a given pair. When this argument is set to ‘1’, the TSE reads only the PSCB.Addr location for 0 branch and the PSCB.addr+2 location for 1 branch. It is assumed that PSCB.Addr+1 and PSCB.Addr+3 are used as a PSCB pair by some other PSCB node. PSCB.Addr.LCBA is ignored and Address will is read in the Control Store using a 1x1 shape. The contents of the PSCB.LCBA in the Shared Memory Pool are set to zero. This argument is stored in the Shared Memory Pool. This argument is valid for NP2G.

Note: This command needs the value of the Search_type field for PSCB shape computation. Since the LUDefTable entry is not read during this command, it uses the content from the local copy of the LUDefCopy_GDH register. The values in the LuDefCopy_GDH register are assumed to be valid prior to this command. This value is stored as a scalar register and is independent for each thread coprocessor.

Table 8-45. RDPSCB_GDH Output Results

Result	Bit Length	Source	Description
PSCB	—	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. For FM: Content of either NPA0 or LCBA0 and NPA1 or LCBA1 will be set to zero (that is, either NPA and LCBA will be valid). Also content of NBT0 will be set to zero when NPA0 is zero and NBT1 will be set to zero when NPA1 is zero (that is, for end node NBT and NPA is not valid). For LPM: No validation is performed based on the actual content of LCBA, NBT, NPA or format fields.
OK/KO	1	Flag	0 KO: Unsuccessful Operation 1 OK: Successful Operation

8.2.7.11 Write PSCB for Fast Table Update (WRPSCB_GDH)

WRPSCB_GDH writes a PSCB stored in PSCBx (in shared memory pool) to the control store. For LPM, the entire PSCB is written to memory to the address given by PSCB.Addr. For FM, the PSCB is written to memory in FMPSCB format. In both cases, WRPSCB_GDH 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 by picocode. This command is not supported for SMT. pscb_cntl/format bits (which defines end node or valid presence of leaf) for the DT/PSCB entry will be computed based on the content of PSCB.LCBA0/1 and no checking/validation is performed, based on the PSCB.NPA0/1 or PSCB.NBT0/1, therefore PSCB.NBT0/1 will be written to memory as is and for LPM PSCB.NPA0/1 is written as is.

Table 8-46. WRPSCB_GDH Input Operands (Page 1 of 2)

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
DPA_PSCB	6	Imm16(5..0)	Imm12(5..0)	The DPA_PSCB is a Full address of the *7.2.4 Shared Memory Pool* on page 172. This indicates that the working address of the PSCB register Full PSCB will occupy 2 quadwords from the shared memory pool. This address must start at an even address.
RdWrPSCB_Cntl	2	Imm16(7..6)	Imm12(7..6)	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
Reserved	1	Imm16(8)	Imm12(8)	Reserved.
PSCB.Addr	26	Shared Memory Pool (121..96)		Address to be written in control store of the PSCB. This is stored in shared memory pool.

Table 8-46. WRPSCB_GDH Input Operands (Page 2 of 2)

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
PSCB.rdwrpscb_LPM _compact_PSCB_en	1	Shared Memory Pool (127)		This optional argument for LPM reduces memory read bandwidth. When not used, it must be set to '0'. This option enables use of the LPM compact PSCB. It indicates the LPM DT/PSCB entry has no valid leaf address and is stored in memory that is 36 bits wide or less than 64 bits wide (that is, Z0 and H1 SRAM memories). It is not used by FM. 0 For FM PSCB/DT and when LPM DT/PSCB is stored in 64-bit or wider memory for width = 1 and height = 1 (DDR DRAM memory). 1 When LPM DT/PSCB is stored in memory that is narrower than 64 bits for width = 1 and height = 1 (36-bit Z0 and H1 SRAM memories) and No Prefix/LCBA is present. When this argument is set to '1', PSCB.Addr+0 and PSCB.Addr+2 are treated as one PSCB pair. PSCB.Addr+1 and PSCB.Addr+3 are treated as separate PSCB pair. Addresses are interleaved for a given pair. When this argument is set to '1', the TSE reads only the PSCB.Addr location for 0 branch and the PSCB.addr+2 location for 1 branch. It is assumed that PSCB.Addr+1 and PSCB.Addr+3 are used as a PSCB pair by some other PSCB node. PSCB.Addr.LCBA is ignored and Address will is read in the Control Store using a 1x1 shape. The contents of the PSCB.LCBA in the Shared Memory Pool are set to zero. This argument is stored in the Shared Memory Pool. This argument is valid for NP2G.
PSCB	—	Shared Memory Pool		The PSCB is read from the Shared Memory Pool PSCB<pn> and written to the Control Store. Some or all of the NPA0, NBT0, LCBA0, NPA1, NBT1, and LCBA1 fields are used. For FM: The contents of NPA0, LCBA0 and NPA1, or LCBA1 are used (that is, either NPA and LCBA is valid). The contents of NBT0 are set to zero when NPA0 is 0, and NBT1 is set to zero when NPA1 is 0 (that is, end-node NBT and NPA are not valid). For LPM: When LCBA contents are zero (that is, no valid leaf/prefix is present), format bits will be set to '00'. No validation is performed based on the content of NBT or NPA fields and written to the Control Store.
Note: This command needs the value of the Search_type field for PSCB shape computation. Since the LuDefTable entry is not read during this command, it uses the content from the local copy of the LUDefCopy_GDH register. The values in the LuDefCopy_GDH register are assumed to be valid prior to this command. This value is stored as a scalar register and is independent for each thread coprocessor.				

Table 8-47. WRPSCB_GDH Output Results

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

8.2.7.12 SetPatBit_GDH

SetPatBit_GDH instruction reads a bit from the HashedKey pattern or Leaf Reference pattern stored in TSRx of the shared memory pool and stores this bit in the SetPatBit_GDH register. Picocode must copy it to the appropriate patbit field of TSEDPA_PSCBx of the shared memory pool.

Table 8-48. SetPatBit_GDH Input Operands

Operand	Bit Length	Operand Source		Description
		Direct	Indirect	
DPA_PSCB	6	Imm16(5..0)	Imm12(5..0)	The DPA_PSCB is a Full address of the thread's shared address (see *7.2.4 Shared Memory Pool* on page 172). This indicates the working address of the PSCB register. The DPA_PSCBx.index field will be used.
TSEDPA_HK_leaf	4	Imm16(9..6)	Imm12(9..6)	Indicates the leaf or HashedKey storage location in the Thread Shared Memory pool. The TSEDPA is the high order 4 bits of the thread's shared memory pool address (see *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary. Use only values of x '8', x 'A', x 'C', and x 'E'. When HK_Leaf_flag = 0, HashedKey will be used for this operation. When HK_Leaf_flag = 1, Leaf content will be used for operation. When Leaf is to be used, Location for leaf will be TSEDPA_HK_leaf; and when HashedKey is used, Location for HashedKey will be TSEDPA_HK_leaf +2.
HK_leaf_flag	1	Imm16(10)	Imm12(10)	When HK_Leaf_flag = 0, HashedKey will be used for this operation and TSEDPA_HK_leaf +2 will be used as a HashedKey Location in shared memory pool. When HK_Leaf_flag = 1, Leaf content will be used for operation and TSEDPA_HK_leaf will be used as a Leaf location in shared memory pool.
HashedKey	192	Shared Memory Pool		Contains the hashed pattern, and it will be stored at TSEDPA_HK_leaf +2 & +3 QW location. Structure of Hashed Key in shared memory pool is shown in the above figure.
HashedKeyLen	8	Shared Memory Pool		Contains length of hashed pattern minus 1, and it will be stored at TSEDPA_HK_leaf +3 (7..0) QW location. Structure of Hashed Key in shared memory pool is shown in the above figure.
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_HK_leaf parameter, for a length of 4 QW.

Table 8-49. SetPatBit_GDH Output Results

Result	Bit Length	Source	Description
SetPatBit_GDH	1	Register	Smallest bit position in which the pattern in HashedKey differs from the pattern in the leaf at TSEDPA_HK_leaf location. Its address in register address map is x'01B'.
SetPatBit_GDH_4bit	4	Register	Smallest four bit positions where the pattern in HashedKey differs from the pattern in leaf at TSEDPA_HK_leaf location. Its address in register address map is x'01C'. This register is valid for *NP2G.*
Note: This command needs the value of the p1p2_max_size value as a part of its computation. Since the LUDefTable entry is not read during this command, it uses the contents from the local copy of the LUDefCopy_GDH register. The values in the LuDefCopy_GDH register are assumed to be valid prior to this command. This value is stored as a scalar register and is independent for each thread coprocessor. SetPatBit_GDH applies to LPM only.			

SetPatBitGDH_4Bit is used by tree management code for multibit NBT LPM search trees and is valid for NP2G.

8.2.8 GTH Hardware Assist Instructions

GTH Hardware Assist instructions can be executed only by the GTH thread (1) on TSE0. General TSE instructions (see *Table 8-26* on page 316) cannot execute concurrently with GTH Hardware Assist instructions.

GTH Hardware Assist instructions are used to perform tree management functions such as insert and delete operations. The general TSE instruction set is also able to perform these functions, but they require additional code steps. For applications requiring a high rate of tree insert and delete operations, the use of the general TSE instruction set is recommended because multiple threads can be employed. Other tree management functions (such as aging, which employs ARDL and TLIR) or memory management functions (which employs TSENQFL and TSDQFL) can be performed using software-created and managed structures.

Table 8-50. General GTH Instructions

Opcode	Command	Detail Section
16	HK_GTH	*8.2.8.1 Hash Key GTH (HK_GTH)* on page 334
17	RDLUDEF_GTH	*8.2.8.2 Read LUDefTable GTH (RDLUDEF GTH)* on page 334
18	TSENQFL	*8.2.8.3 Tree Search Enqueue Free List (TSENQFL)* on page 335
19	TSDQFL	*8.2.8.4 Tree Search Dequeue Free List (TSDQFL)* on page 336
20	RCLR	*8.2.8.5 Read Current Leaf from Rope (RCLR)* on page 336
21	ARDL	*8.2.8.6 Advance Rope with Optional Delete Leaf (ARDL)* on page 337
22	TLIR	*8.2.8.7 Tree Leaf Insert Rope (TLIR)* on page 338
23	Reserved	
24	CLRPSCB	*8.2.8.8 Clear PSCB (CLRPSCB)* on page 338
25	RDPSCB	*8.2.8.9 Read PSCB (RDPSCB)* on page 339
26	WRPSCB	*8.2.8.10 Write PSCB (WRPSCB)* on page 339
27	PUSHPSCB	*8.2.8.11 Push PSCB (PUSHPSCB)* on page 340
28	DISTPOS	*8.2.8.12 Distinguish (DISTPOS)* on page 340
29	TSR0PAT	*8.2.8.13 TSR0 Pattern (TSR0PAT)* on page 341
30	PAT2DTA	*8.2.8.14 Pattern 2DTA (PAT2DTA)* on page 342
31	Reserved	
Note: The instructions listed in this table can only be executed by the GTH.		

8.2.8.1 Hash Key GTH (HK_GTH)

Table 8-51. 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 *7.2.4 Shared Memory Pool* on page 172) 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 8-52. 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 293 and Section *8.2.2 Input Key and Color Register for SMT Trees* on page 293. 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 8-53. 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 8-54. 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 8-55. 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 GTH only) 101 PSCB1.Addr (for GTH only) 110 PSCB2.Addr (for GTH only)
LCBA0 / 1 PSCB0 / 1 / 2.Addr	26	Register		The address to be freed or enqueued

Table 8-56. 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 8-57. 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 GFH only) 101 PSCB1.Addr (for GFH only) 110 PSCB2.Addr (for GFH only)

Table 8-58. 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 8-59. 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 *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary.

Table 8-60. 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 8-61. 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 *7.2.4 Shared Memory Pool* on page 172) 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 8-62. 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 8-63. 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 *7.2.4 Shared Memory Pool* on page 172) 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 8-64. 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 8-65. CLRPSCB Input Operands

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

Table 8-66. 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 8-67. 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 8-68. 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 8-69. 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 8-70. PUSHPSCB Input Operands

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

Table 8-71. 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 8-72. 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 *7.2.4 Shared Memory Pool* on page 172) 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 8-73. 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 8-74. 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 *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary. Use only values of x'8', x'A', x'C', and x'E'.

Table 8-75. 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 8-76. 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 *7.2.4 Shared Memory Pool* on page 172) and is constrained to be on a four-QW boundary. Use only values of x'8', X'A', X'C', and X'E'.

Table 8-77. 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 8-78. 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 8-14: 192-Bit IP Hash Function* on page 343.
2	192-bit MAC Hash	See *Figure 8-15: MAC Hash Function* on page 344.
3	192-bit Network DISTPATCH	See *Figure 8-16: Network Dispatcher Hash Function* on page 345.
4	Reserved	
5	48-bit MAC swap	See *Figure 8-17: 48-Bit MAC Hash Function* on page 346.
6	60-bit MAC swap	See *Figure 8-18: 60-Bit MAC Hash Function* on page 347.
7	Reserved	
8	8-bit Hasher	See *Figure 8-19: 8-bit Hash Function* on page 348.
9	12-bit Hasher	See *Figure 8-20: 12-bit Hash Function* on page 348.
10	16-bit Hasher	See *Figure 8-21: 16 bit Hash Function* on page 349.
11-15	Reserved	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. The Hash_type field of the LUDefTable defines the hasher to be used. If color is enabled, the color is inserted at the bit position given by DT_Size in the LUDefTable and 16 least significant bits of the key are ignored, since a maximum key length of 176 bits is supported when color is enabled.

Figure 8-13. No-Hash Function

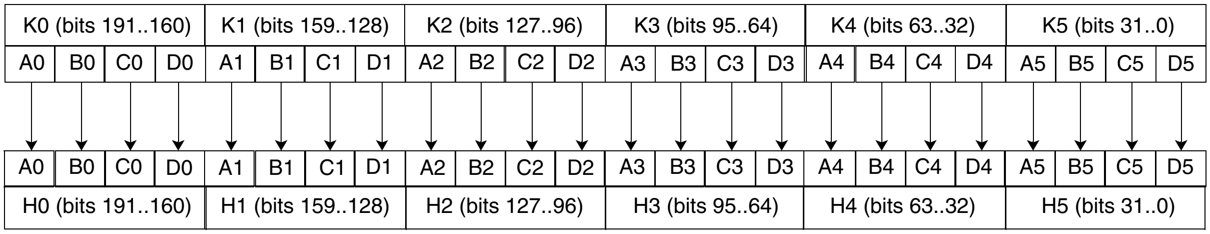


Figure 8-14. 192-Bit IP Hash Function

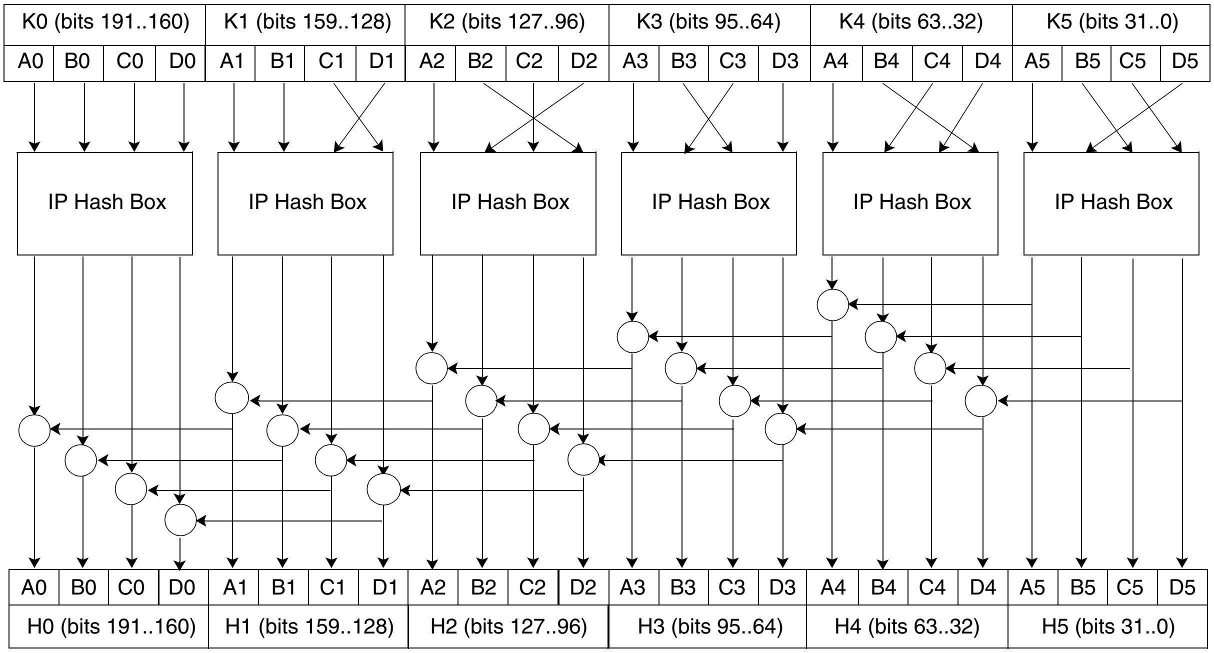


Figure 8-15. MAC Hash Function

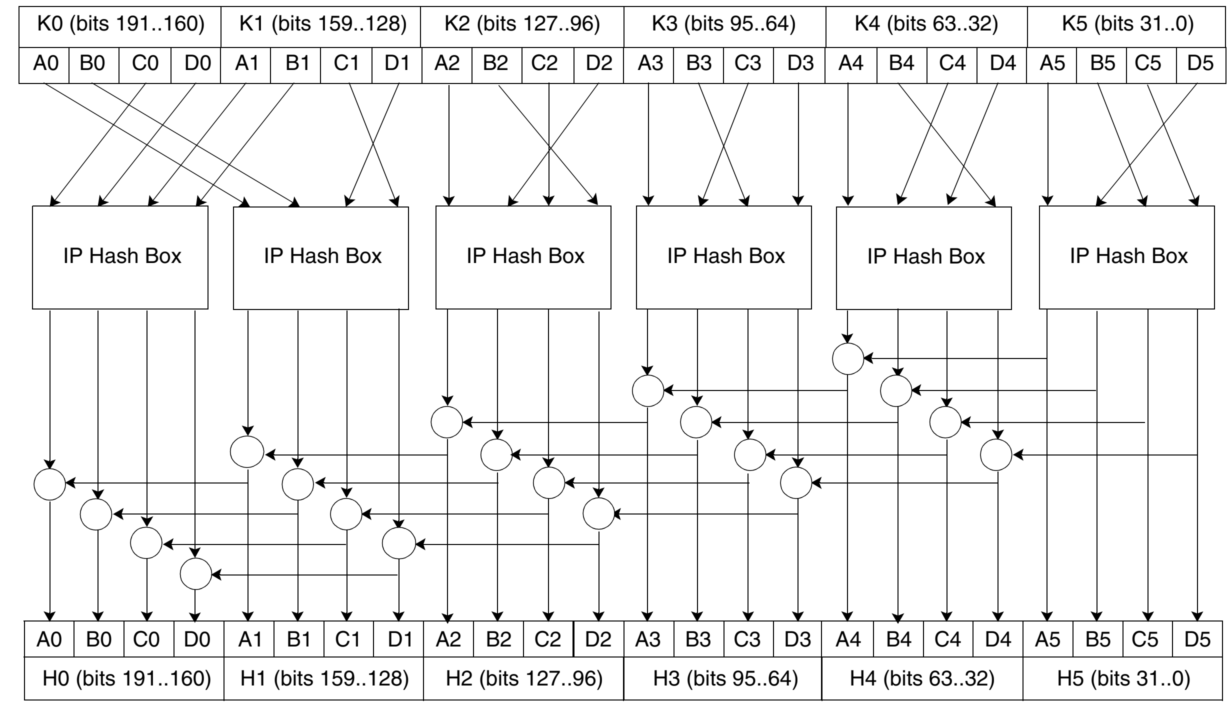


Figure 8-16. Network Dispatcher Hash Function

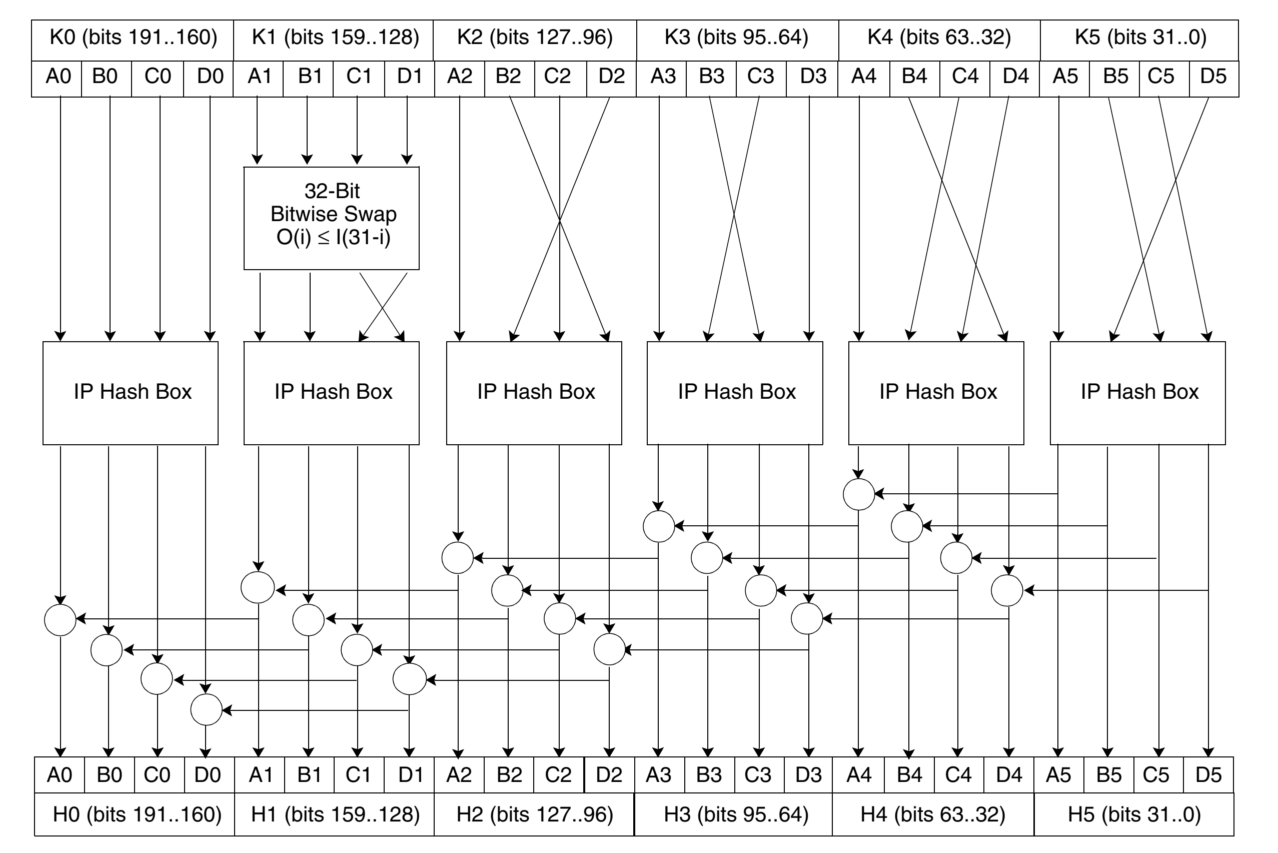


Figure 8-17. 48-Bit MAC Hash Function

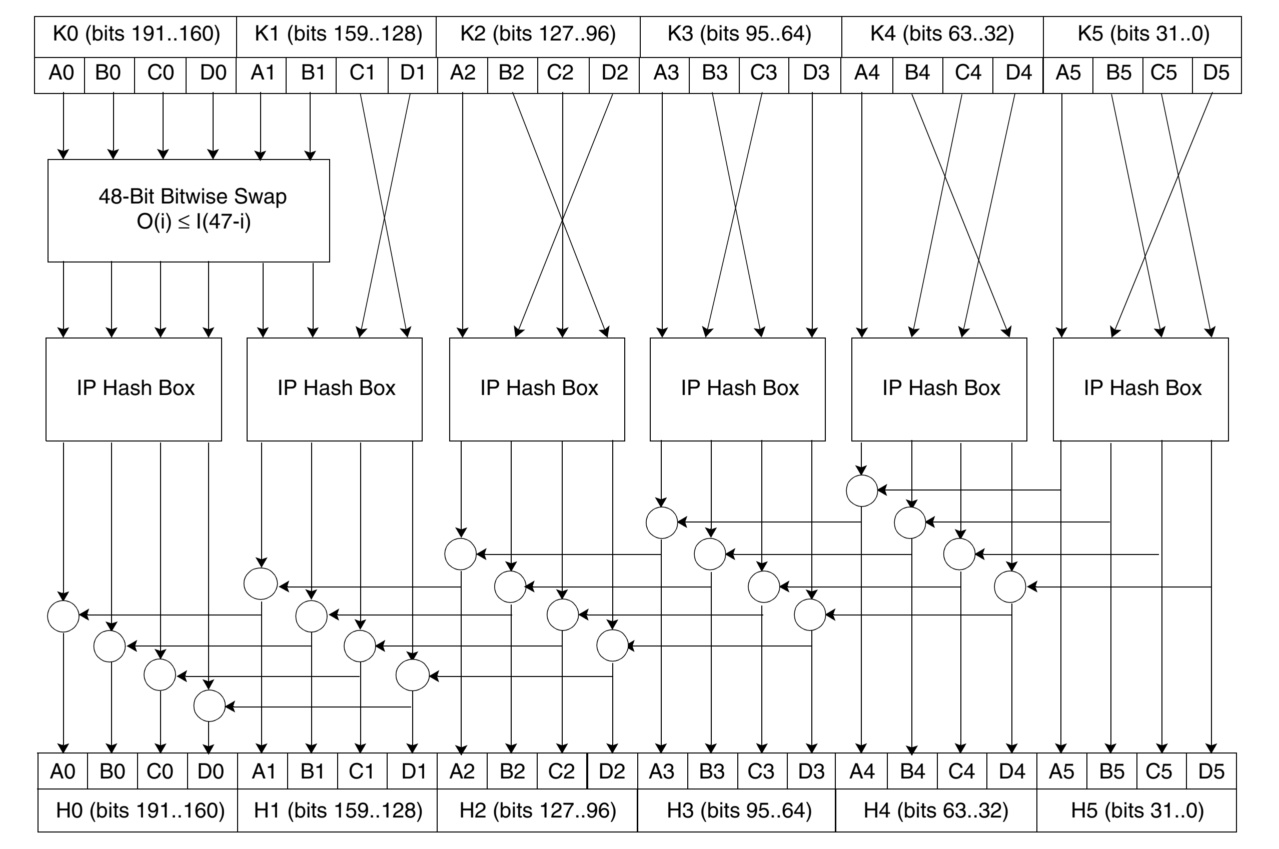


Figure 8-18. 60-Bit MAC Hash Function

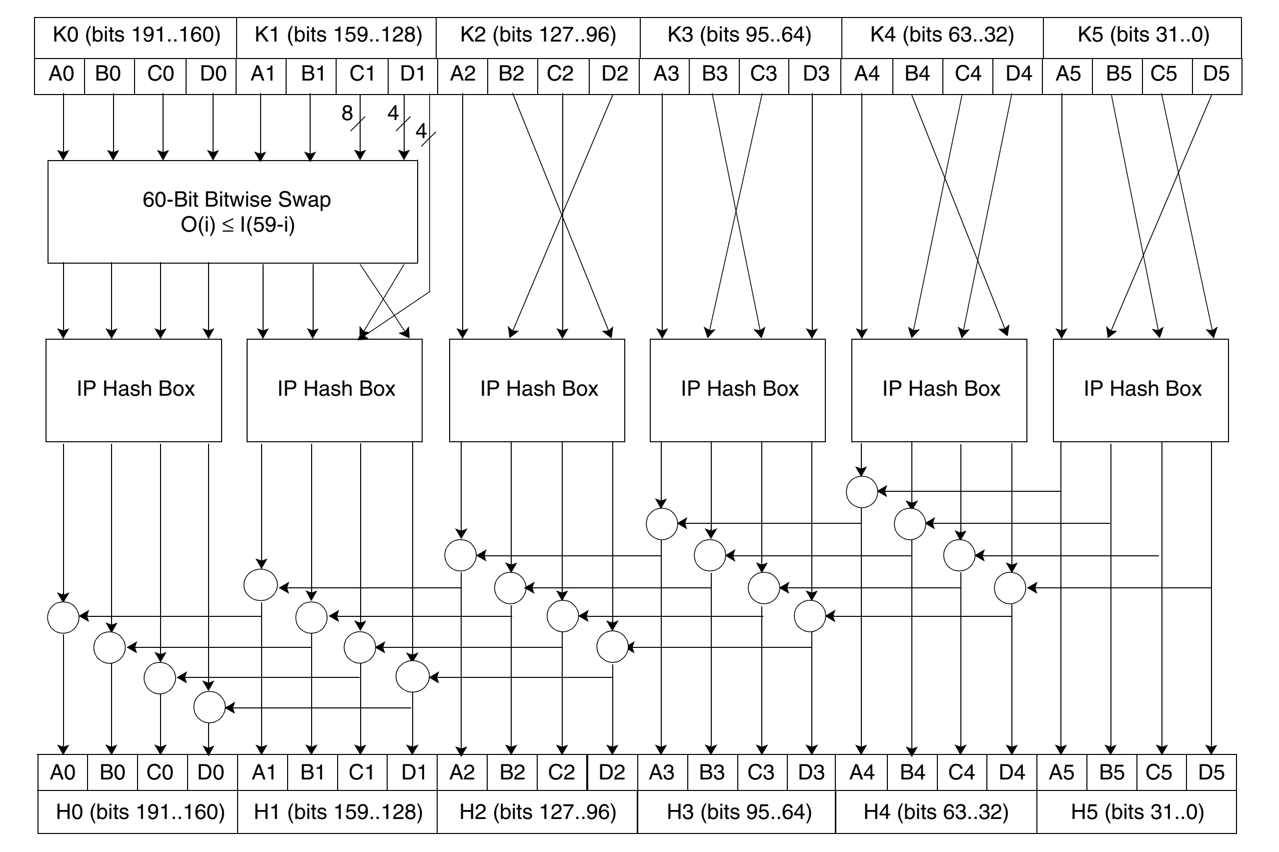
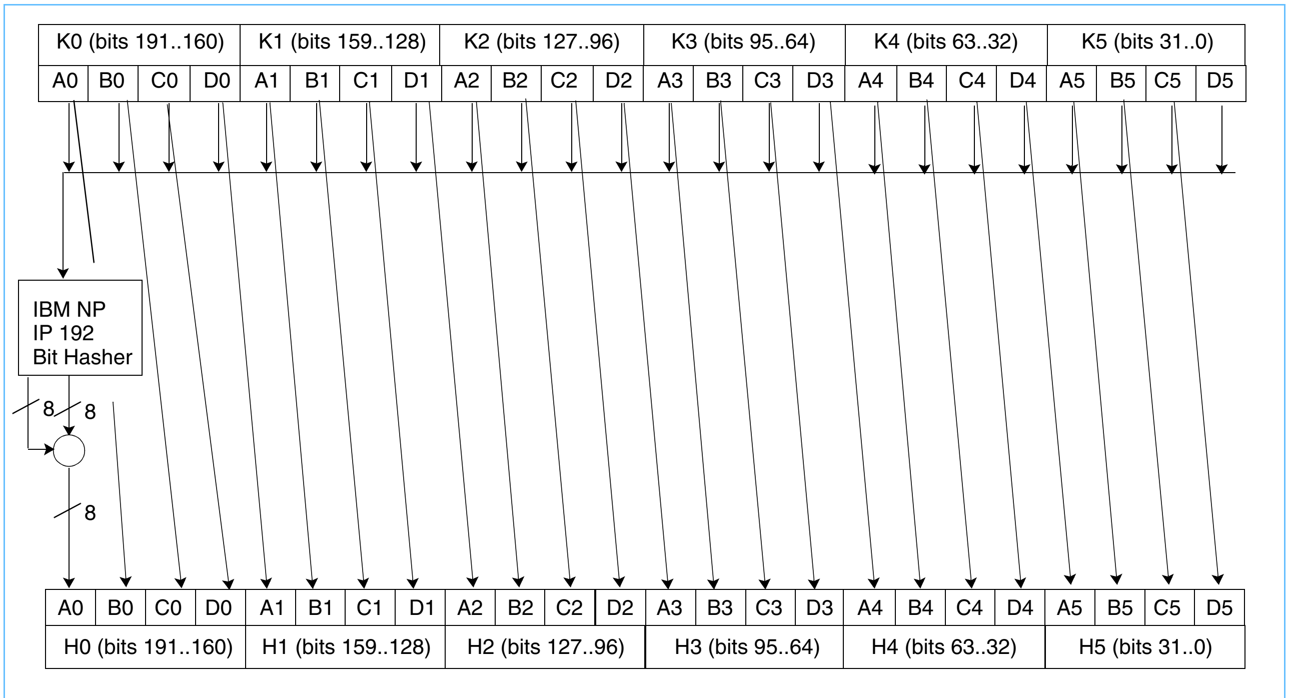
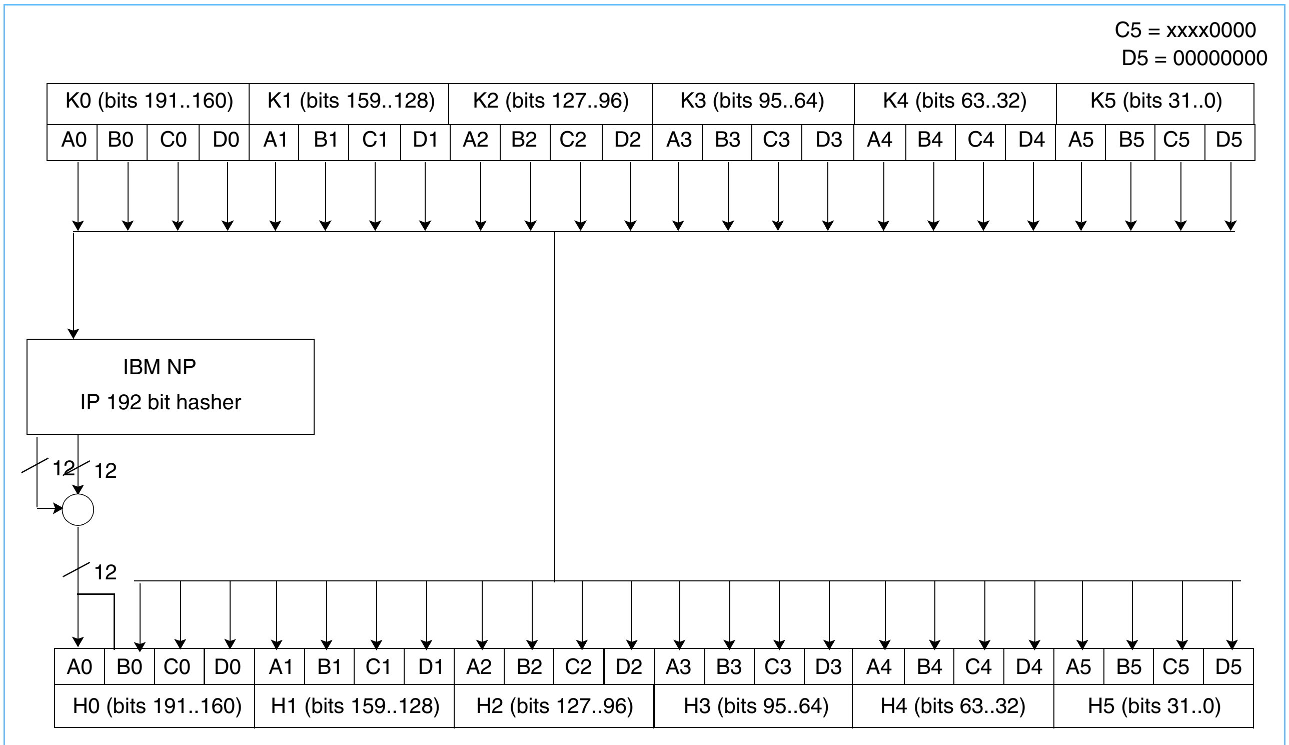


Figure 8-19. 8-bit Hash Function



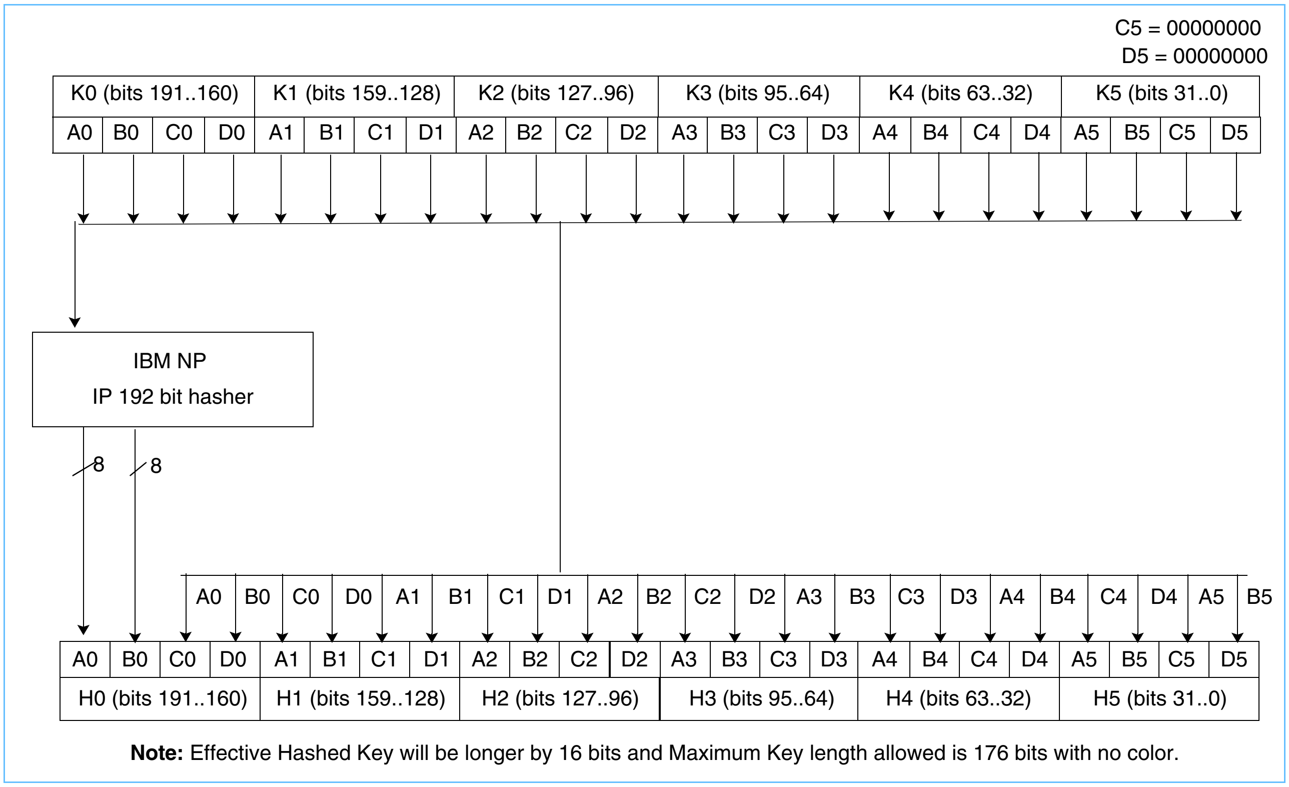
Note: Effective Hashed Key will be longer by 8 bits and Maximum Key length allowed is 184 bits with no color.

Figure 8-20. 12-bit Hash Function



Note: Effective Hashed Key will be longer by 12 bits and Maximum Key length allowed is 180 bits with no color.

Figure 8-21. 16 bit Hash Function



Note: Effective Hashed Key will be longer by 16 bits and Maximum Key length allowed is 176 bits with no color.

9. Serial / Parallel Manager Interface

Located within the Embedded Processor Complex (EPC), the Serial / Parallel Manager (SPM) Interface is a serial interface for communication 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 NP2G is the master and the external SPM module is the only slave[1]. The SPM Interface loads picocode, allowing management of physical layer devices (PHYs) and access to card-based functions such as light-emitting diodes (LEDs).

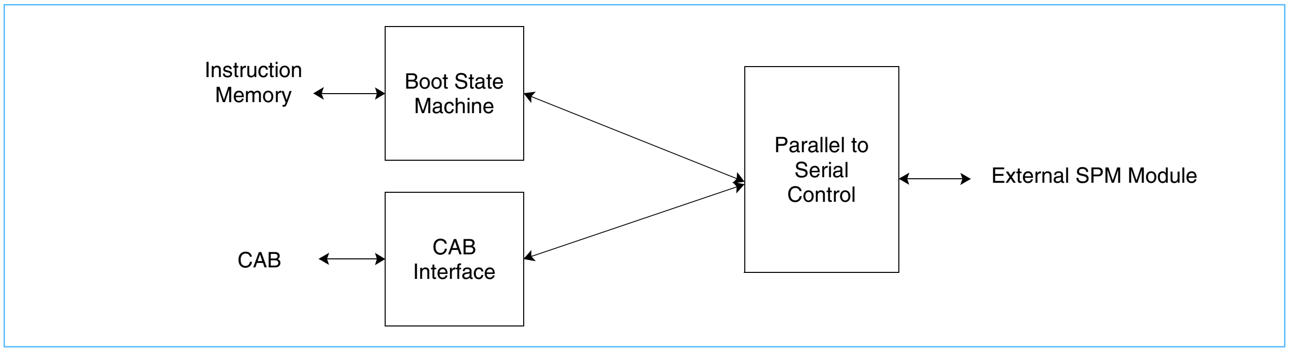
The SPM Interface supports:

- An external SPM module
- Boot code load via external SPM and EEPROM
- Boot override via CABWatch interface or Boot_Picocode configuration device I/O
- Access to external PHYs, LEDs, management, and card-based functions

9.1 SPM Interface Components

Figure 9-1 shows the functional blocks of the SPM Interface. The list following the figure describes them.

Figure 9-1. SPM Interface Block Diagram



Boot State Machine — Starts up after reset if configured to do so by an external I/O 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 Guided Frame Handler (GFH) thread, which executes the loaded code.

CAB Interface — A memory mapped interface to the NP2G 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.

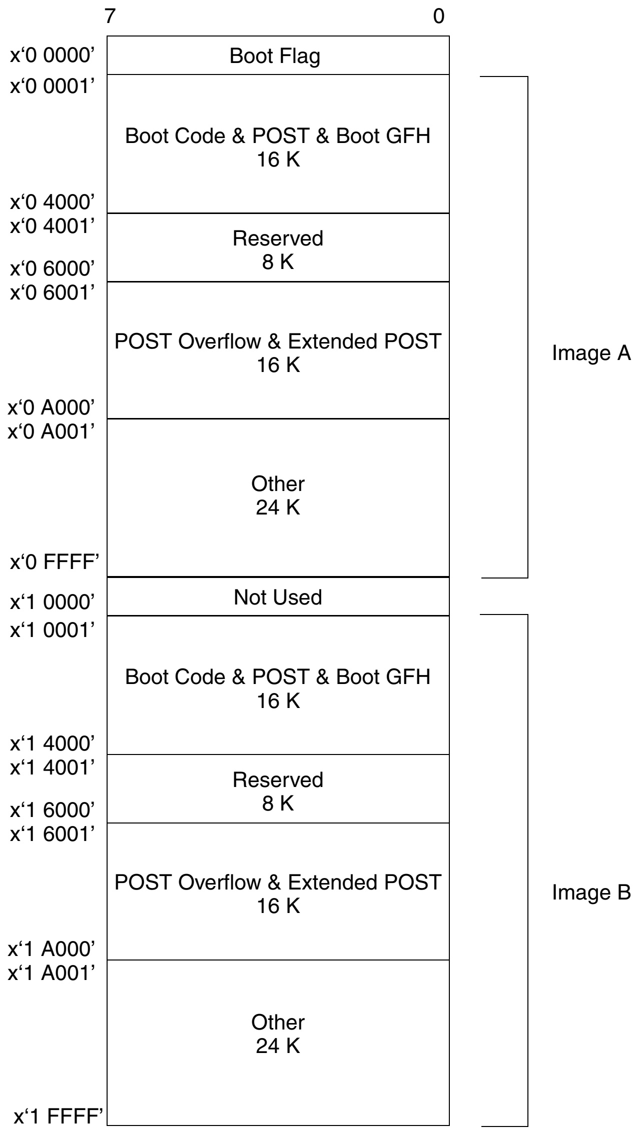
[1] IBM does not supply the external SPM module.

9.2 SPM Interface Data Flow

The SPM Interface is used initially to boot the NP2G. 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 causes a POR interrupt that 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.

The Boot State Machine supports two images of boot code in external EEPROM (see *Figure 9-2*). 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.

Figure 9-2. EPC Boot Image in External EEPROM

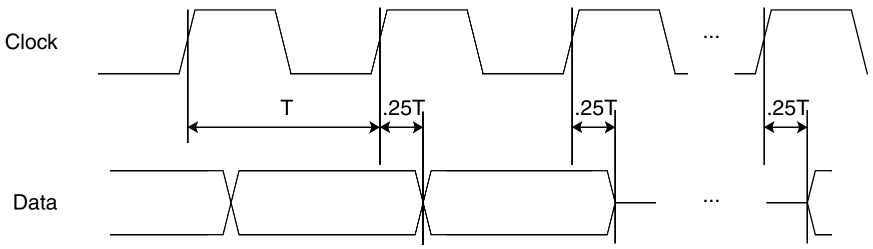


The SPM Interface is also used during initialization and statistics gathering. It interfaces with the Ethernet PHYs, card LEDs, and other card-level structures through an external SPM interface module supplied by the customer. These external structures are mapped to the Control Access Bus (CAB) address 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 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 NP2G and can be one to four bytes in length. *Figure 9-3* illustrates the timing of the SPM Interface.

Figure 9-3. SPM Bit Timing



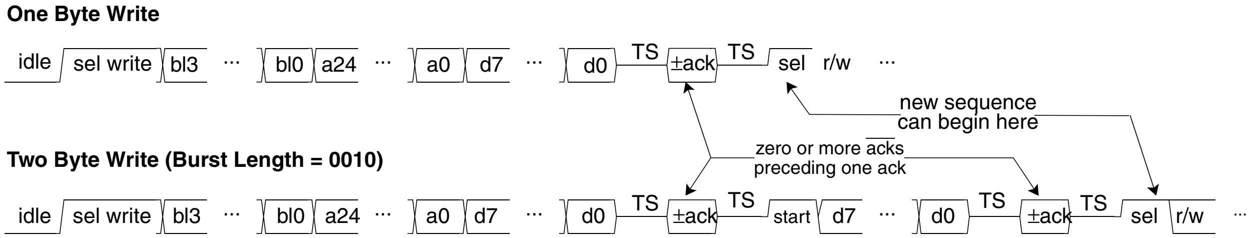
For single-byte transfers, the exchange begins when the NP2G drives the data signal to ‘1’ for one clock period. This “select” indication is followed by a 1-bit write/$\overline{\text{read}}$ indication, a 4-bit burst length indication, and a 25-bit address value.

For read and write transfers, the SPM Interface master waits for a response from the SPM Interface slave, and the slave communicates with the master using the following acknowledgments:

- ack: a ‘1’ driven onto the data bus by the SPM Interface slave to indicate each successful byte operation, either read or write.
- $\overline{\text{ack}}$: a ‘0’ driven onto the data bus by the SPM Interface slave while the byte operation is in progress.

For write transfers (see *Figure 9-4*), the address is followed immediately by eight bits of data. The NP2G 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 NP2G 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.

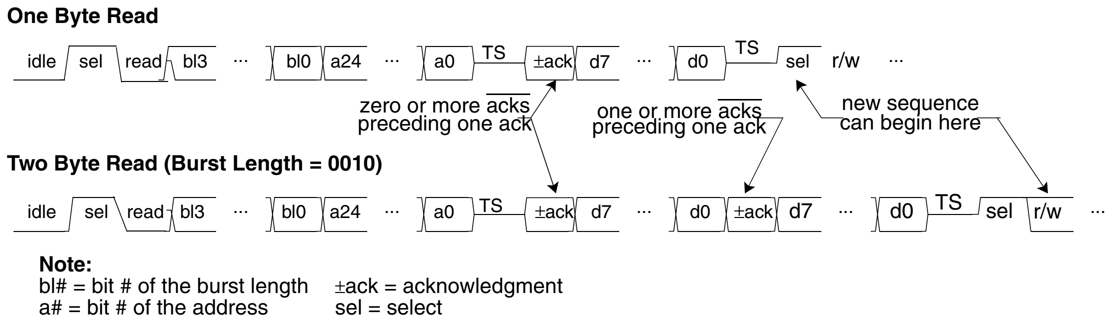
Figure 9-4. SPM Interface Write Protocol



Note:
bl# = bit # of the burst length
a# = bit # of the address
±ack = acknowledgment
sel = select

For read transfers (see *Figure 9-5*), the address is followed by the NP2G 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 NP2G 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.

Figure 9-5. SPM Interface Read Protocol


One Byte Read
idle
sel
read
bl3
···
bl0
a24
···
a0
TS
±ack
d7
···
d0
TS
sel
r/w
···
zero or more $\overline{\text{acks}}$ preceding one ack
one or more $\overline{\text{acks}}$ preceding one ack
new sequence can begin here
Two Byte Read (Burst Length = 0010)
idle
sel
read
bl3
···
bl0
a24
···
a0
TS
±ack
d7
···
d0
±ack
d7
···
d0
TS
sel
r/w
···
Note:
bl# = bit # of the burst length
a# = bit # of the address
±ack = acknowledgment
sel = select

The protocol for multiple byte transfers is similar, except that each byte written is accompanied by a bus turn-around, 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 data bus followed by one ‘1’ on the data bus (ack), and immediately followed by the eight bits of data.

9.4 SPM CAB Address Space

The SPM Interface enables the control access bus (CAB) to access an external EEPROM and other devices attached via an external SPM module developed by the customer. The address space is divided into three areas:

- Byte access
- Word access
- EEPROM access

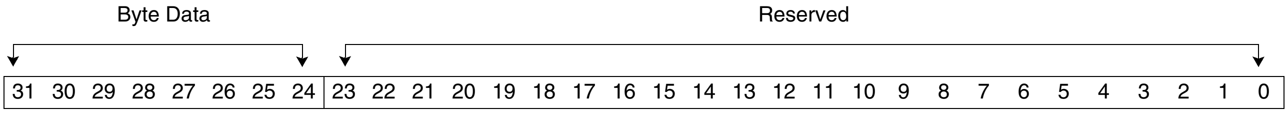
9.4.1 Byte Access Space

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’



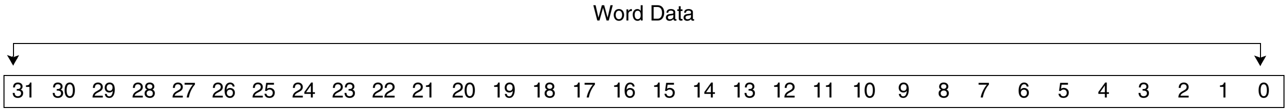
Field Name	Bit(s)	Reset	Description
Byte Data	31:24		Data at this location
Reserved	23:0		Reserved

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 external SPM module can be combined to provide access to an attached EEPROM. The EEPROM access space contains locations for 16 M 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 9-1*.

Table 9-1. 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 Single-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’

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

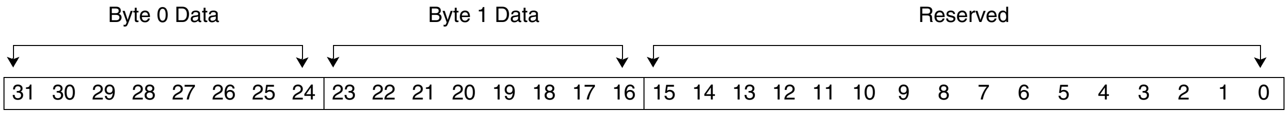
Field Name	Bit(s)	Reset	Description
Byte Data	31:24		Data at starting byte address
Reserved	23:0		Reserved

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’



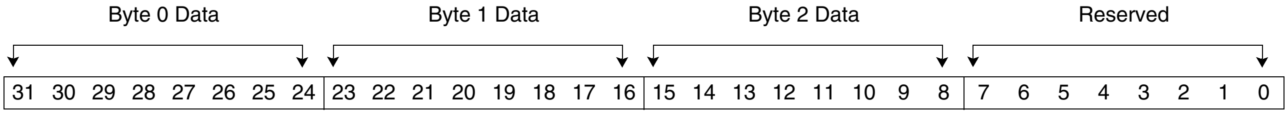
Field Name	Bit(s)	Reset	Description
Byte 0 Data	31:24		Data at starting byte address
Byte 1 Data	23:16		Data at starting byte address + 1
Reserved	15:0		Reserved

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’

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

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™ Subsystem

10.1 Description

The NP2G 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 NP2G.

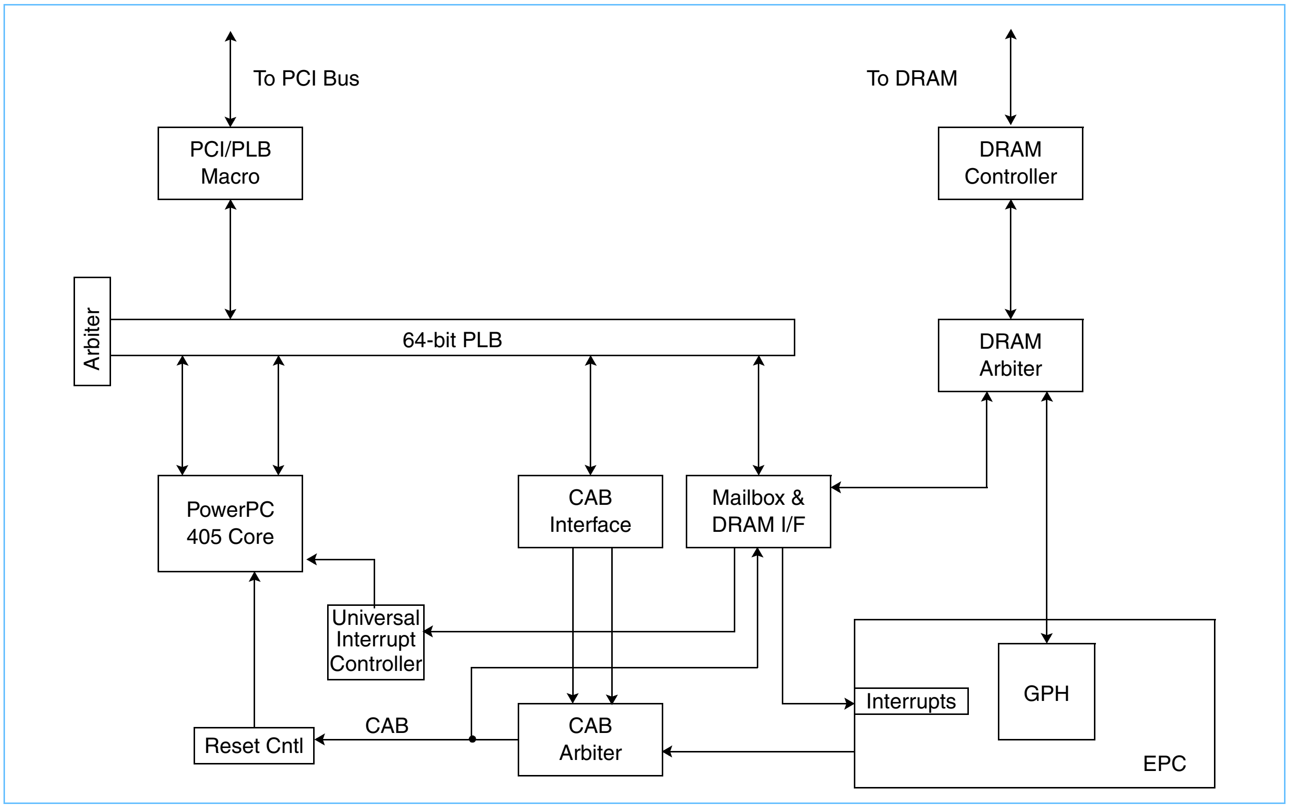
Standard IBM PowerPC macros include the following:

- 133 MHz PPC405 Processor Core with 16 K of instruction cache and 16 K of data cache
  - NP2G uses the PPC405D4V6 core
- 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

Documentation for the above macros is contained in the *IBM PowerPC 405GP Embedded Processor User's Manual* (http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405GP_Embedded_Processor) and is not repeated here. That document also contains information for other macro library components not contained in the NP2G which does not apply to the above macros.

The embedded PowerPC subsystem includes a CAB Interface PLB slave unit to access NP2G internal structures and a Mailbox and DRAM Interface PLB slave unit for inter-processor communications and access to PowerPC instructions.

Figure 10-1. PowerPC Subsystem 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.

10.2.1 Processor Local Bus (PLB)

The PLB is the high performance bus used to access memory, PCI devices, and NP2G structures through the PLB interface units. The PLB interface units shown in *Figure 10-1*, 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 10-1. 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

10.2.2 Device Control Register (DCR) Bus

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.

Table 10-2. Device Control Registers

Register	DCR Offset	Description	Access
Processor Local Bus Arbiter Macro			
REV	x‘02’	PLB Arbiter Revision ID	R
BESR	x‘04’	PLB Bus Error Status Register	R/Clear
BEAR	x‘06’	PLB Bus Error Address Register	R
ACR	x‘07’	PLB Arbiter Control Register	R/W
Universal Interrupt Controller Macro			
SR	x‘00’	UIC Status Register	R/Clear
SRS	x‘01’	UIC Status Register Set	R/Set
ER	x‘02’	UIC Enable Register	R/W
CR	x‘03’	UIC Critical Register	R/W
PR	x‘04’	UIC Polarity Register	R/W
TR	x‘05’	UIC Triggering Register	R/W
MSR	x‘06’	UIC Masked Status Register	R
VR	x‘07’	UIC Vector Register	R
VCR	x‘08’	UIC Vector Configuration Register	W

All PLB arbiter registers and Universal Interrupt Controller 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 *IBM PowerPC 405GP Embedded Processor User’s Manual* (http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405GP_Embedded_Processor).

10.3 PLB Address Map

Components of the embedded PowerPC subsystem 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.

Table 10-3. Address Map for CAB Interface Macro

Symbolic Address	PLB Address	Description	Access
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[1]
PwrPC_CAB_Status	x‘7800 0018’	PowerPC CAB Status Register	R
PwrPC_CAB_Mask	x’7800 0020’	PowerPC CAB Mask Register	R/W
PwrPC_CAB_WUM_Data	x’7800 0028’	PowerPC CAB Write Under Mask Data Register	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[1]
Host_CAB_Status	x‘7800 8018’	PCI Host CAB Status Register	R
Host_CAB_Mask	x‘7800 8020’	PCI Host CAB Mask Register	R/W
Host_CAB_WUM_Data	x’7800 8028’	PCI Host CAB Write Under Mask Data Register	W

Unassigned addresses in the range x‘7801 0000’ - x‘7801 FFFF’ are reserved.

1. Additional action occurs on register access using the specified address. Refer to register detailed section for more information.

Table 10-4. PLB Address Map for Mailbox and DRAM Interface Macro (Page 1 of 2)

Symbolic Address	PLB Address	Description	Access
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 Subsystem to PCI Host Resource Register	R/W[1]
P2H_Msg_Addr	x‘7801 0018’	PowerPC Subsystem to PCI Host Message Address Register	R/W[1]
P2H_Doorbell	x‘7801 0020’	PowerPC Subsystem to PCI Host Doorbell Register (PowerPC Access)	R/SUM
	x‘7801 0028’	PowerPC Subsystem to PCI Host Doorbell Register (PCI Host Access)	R/RUM
H2P_Msg_Addr	x‘7801 0050’	PCI Host to PowerPC Subsystem Message Address Register (Reset Status)	R[1]
	x‘7801 0060’	PCI Host to PowerPC Subsystem Message Address Register	R/W[1]
H2P_Doorbell	x‘7801 0030’	PCI Host to PowerPC Subsystem 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 Subsystem Message Resource Register	R/W

Unassigned addresses in the range x‘7801 0000’ - x‘7801 FFFF’ are reserved

1. Additional action occurs on register access using the specified address. Refer to register detailed section for more information.

Table 10-4. PLB Address Map for Mailbox and DRAM Interface Macro (Page 2 of 2)

Symbolic Address	PLB Address	Description	Access
E2P_Msg_Addr	x'7801 0048'	EPC to PowerPC Subsystem Message Address Register	R[1]
E2P_Doorbell	x'7801 0058'	EPC to PowerPC Subsystem Doorbell Register (PowerPC Access)	R/RUM
P2E_Msg_Addr	x'7801 0068'	PowerPC Subsystem to EPC Message Address Register	R/W
P2E_Doorbell	x'7801 0070'	PowerPC Subsystem 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[1]
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
SESR	x'7801 00C0'	Slave Error Status Register	RWR[1]
Perr_Cntr	x'7801 00C8'	Parity Error Counter Register	R
PwrPC_Inst_Store	x'0000 0000' - x'07FF FFFF'	PowerPC Instruction/Data DRAM	R/W

Unassigned addresses in the range x'7801 0000' - x'7801 FFFF' are reserved

1. Additional action occurs on register access using the specified address. Refer to register detailed section for more information.

10.4 CAB Address Map

Some components of the embedded PowerPC subsystem are also accessible via the NP2G's CAB Interface. These components are accessed using CAB addresses as shown in the CAB Address Map.

Table 10-5. CAB Address Map for Mailbox and DRAM Interface Macro

Symbolic Address	CAB Address	Description	Access
Boot_Redir_Inst	x'3800 0110' - x'3800 0117'	Boot Redirection Instruction Registers for instruction addresses x'FFFF FFE0' - x'FFFF FFFC'	R/W
PwrPC_Mach_Chk	x'3800 0210'	PowerPC Subsystem Machine Check Register	R
E2P_Msg_Resource	x'3801 0010'	EPC to PowerPC Subsystem Message Resource Register	R[1]
E2P_Msg_Addr	x'3801 0020'	EPC to PowerPC Subsystem 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 Subsystem to EPC Message Address Register	R
	x'3802 0010'	PowerPC Subsystem to EPC Message Address Register	R[1]
P2E_Doorbell	x'3801 0100'	PowerPC Subsystem to EPC Doorbell Register (PowerPC Access)	R/RUM
E2H_Msg_Resource	x'3801 0200'	EPC to PCI Host Message Resource Register	R[1]
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[1]
	x'3802 0020'	PCI Host to EPC Message Address Register	R[1]
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

1. Additional action occurs on register access using the specified address. Refer to register detailed section for more information.

10.5 Universal Interrupt Controller (UIC) Macro

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 10-6*.

Table 10-6. UIC Interrupt Assignments

Interrupt	Polarity	Sensitivity	Interrupt Source
0	High	Edge	DRAM D6 Parity Error
1	Programmable	Programmable	External Interrupt 0 (PCI_Bus_NM_Int input pin)
2	Programmable	Programmable	External Interrupt 1 (PCI_Bus_M_Int input pin)
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 generated when an external PCI master writes to the PCI Command Register or bit 13 of the Bridge Options 2 Register is set to ‘1’ via PCI configuration. See description of the PCI macro’s Bridge Options 2 Register in the PPC405GP Embedded Controller User’s Manual for details.
8-31	High	Level	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 10-6. For details regarding the control of the UIC, including the programming of interrupts, see the *IBM PowerPC 405GP Embedded Processor User’s Manual* (http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405GP_Embedded_Processor

10.6 PCI/PLB Bridge 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).

Values of the PCI Device Configuration Header for the are initialized by hardware. These values are shown in *Table 10-7*. When the boot picocode is loaded from the Management Bus (Boot_Picocode is tied to '0') and the PowerPC subsystem boots from DRAM D6 (Boot_PPC is tied to '0', see *2.1.8 Miscellaneous Pins* on page 75), a general reset initializes the PCI/PLB macro's Bridge Options 2 Register (PCIBRDGOPT2) with its Host Configuration Enable bit set to '0' (disabled). This allows the PowerPC subsystem to alter the contents of the PCI Device Configuration Header registers prior to access by external configuration. The PowerPC code then enables external host configuration.

Table 10-7. NP2G PCI Device Configuration Header Values

Register Name	Register Value
Vendor ID	x'1014'
Device ID	x'01E8'
Revision ID	x'00'
Class Code	x'028000'
Subsystem ID	x'0000"
Subsystem Vendor ID	x'0000'

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 10-8* shows the address map from the view of the PLB, that is, as decoded by the PCI/PLB macro as a PLB slave.

Table 10-8. PLB Address Map for PCI/PLB Macro (Page 1 of 2)

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' (I/O)
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' (I/O)
x'EC00 0000' - x'EEBF FFFF'	Reserved PCI/PLB Macro does not respond	
x'EEC0 0000' - x'EECF FFFF'	PCI Configuration Address and Data x'EEC0 0000': PCICFGADDR x'EEC0 0004': PCICFGDATA x'EEC0 0008' - x'EECF FFFF': Reserved (can mirror PCICFGDATA)	

Table 10-8. PLB Address Map for PCI/PLB Macro (Page 2 of 2)

PLB Address Range	Description	PCI Address Range
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.	
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’: PTM2MS x‘EF40 003C’: PTM2LA 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 NP2G, 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 to x‘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 to x‘07FF 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.

Unless the boot picocode is loaded from the SPM (Boot_Picocode tied to ‘0’) and the PowerPC subsystem boots from DRAM D6 (Boot_PPC tied to ‘0’), a 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. Otherwise, PowerPC code must enable external host configuration and may alter the contents of the PCI Device Configuration Header registers prior to enabling external host configuration.

Table 10-9. PCI/PLB Bridge Macro Configuration Registers (Page 1 of 2)

Register	Offset	Description	Access	
			Local	Host
VENDID	x‘01’ - x‘00’	PCI Vendor ID	R/W	R
DEVID	x‘03’ - x‘02’	PCI Device ID	R/W	R
CMD	x‘05’ - x‘04’	PCI Command Register	R/W	R/W
STATUS	x‘07’ - x‘06’	PCI Status Register	R/W	R/W
REVID	x‘08’	PCI Revision ID	R/W	R
CLS	x‘0B’ - x‘09’	PCI Class Register	R/W	R
CACHELS	x‘0C’	PCI Cache Line Size	R	R
LATTIM	x‘0D’	PCI Latency Timer	R/W	R/W
HDTYPE	x‘0E’	PCI Header Type	R	R
BIST	x‘0F’	PCI Built in Self Test Control	R	R
PTM1BAR	x‘17’ - x‘14’	PCI PTM 1 BAR	R/W	R/W
PTM2BAR	x‘1B’ - x‘18’	PCI PTM 2 BAR	R/W	R/W
SBSYSVID	x‘2D’ - x‘2C’	PCI Subsystem Vendor ID	R/W	R
SBSYSID	x‘2F’ - x‘2E’	PCI Subsystem ID	R/W	R
CAP	x‘34’	PCI Capabilities Pointer	R	R
INTLN	x‘3C’	PCI Interrupt Line	R/W	R/W
INTPN	x‘3D’	PCI Interrupt Pin	R	R
MINGNT	x‘3E’	PCI Minimum Grant	R	R
MAXLTNCY	x‘3F’	PCI Maximum Latency	R	R
ICS	x‘44’	PCI Interrupt Control/Status	R/W	R/W
ERREN	x‘48’	Error Enable	R/W	R/W
ERRSTS	x‘49’	Error Status	R/W	R/W
BRDGOPT1	x‘4B’ - x‘4A’	PCI Bridge Options 1	R/W	R/W
PLBBESR0	x‘4F’ - x‘4C’	PCI Slave Error Syndrome 0	R/W	R/W
PLBBESR1	x‘53’ - x‘50’	PCI Slave Error Syndrome 1	R/W	R/W
PLBBEAR	x‘57’ - x‘54’	PCI Slave Error Address Register	R	R
CAPID	x‘58’	Capability Identifier	R	R

Table 10-9. PCI/PLB Bridge Macro Configuration Registers (Page 2 of 2)

Register	Offset	Description	Access	
			Local	Host
NEXTIPTR	x‘59’	Next Item Pointer	R	R
PMC	x‘5B’ - x‘5A’	Power Management Capabilities	R	R
PMCSR	x‘5D’ - x‘5C’	Power Management Control Status	R/W	R/W
PMCSRBSE	x‘5E’	PMCSR PCI to PCI Bridge Support Extensions	R	R
DATA	x‘5F’	Data	R	R
BRDGOPT2	x‘63’ - x‘60’	PCI Bridge Options 2	R/W	R/W
PMSCRR	x‘64’	Power Management State Change Request Register	R/W	R/W

For further details regarding the PCI/PLB macro’s control and configuration registers, see the *IBM PowerPC 405GP Embedded Processor User’s Manual* (http://www-3.ibm.com/chips/techlib/techlib.nsf/products/PowerPC_405GP_Embedded_Processor).

10.7 CAB Interface Macro

The CAB Interface macro provides duplicate facilities to support independent CAB access of IBM Network Processor control and status facilities by the PCI Host processor and the embedded PowerPC subsystem. Exclusive access to these facilities, if required, is enforced through software discipline.

The PCI Host Processor can access the IBM Network Processor’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). If the CAB Data register is written, then the CAB access will be a write access. Likewise reading the CAB Data register will result in a CAB read access.
- Status register is read to determine, for polled access, whether read data is ready (rd_rdy).

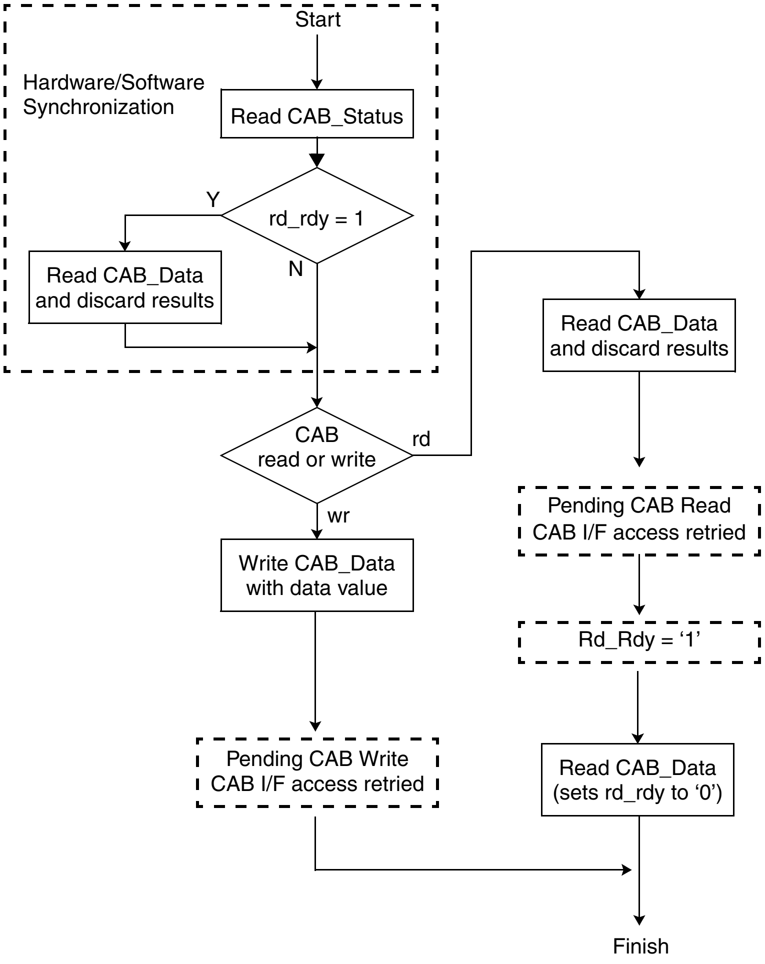
The CAB Control register ($w/\overline{p}$) controls the two modes of PLB protocol for CAB access:

- 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.
- CAB access in polled mode requires software to follow the protocol defined in *Figure 10-2: Polled Access Flow Diagram* on page 373. Behavior for CAB accesses not following this protocol is undefined and may result in adverse effects. Polled access, $w/\overline{p}$ = ‘0’ requires software to read the CAB Status register to synchronize software with the hardware when performing a CAB transaction. A CAB read transaction requires at least two read accesses to the CAB_Data register. An additional read access of the CAB_Data register may be required if the software is not synchronized with the hardware. Software synchronization is determined initially by reading the CAB_Status register. The software is not synchronized when the rd_rdy status bit is set to ‘1’. Synchronization is achieved by reading the CAB_Data register and discarding the result.

  A subsequent read of the CAB Data register initiates a CAB read access from the CAB address contained in the CAB Address register. The data returned as a result of this read of the CAB_Data register is stale data and is discarded by software. Software must then perform a final read of the CAB_Data register to acquire the data accessed on the CAB and return the CAB Interface to its starting state (rd_rdy = ‘0’).

  The NP2G causes subsequent accesses to the CAB interface registers to be retried until a pending CAB access is complete. Prolonged locking of the CAB by the EPC will result in extended retries by the PowerPC subsystem or PCI Host. Under these conditions, system hardware and software design must be able to tolerate long periods of retry.

Figure 10-2. Polled Access Flow Diagram

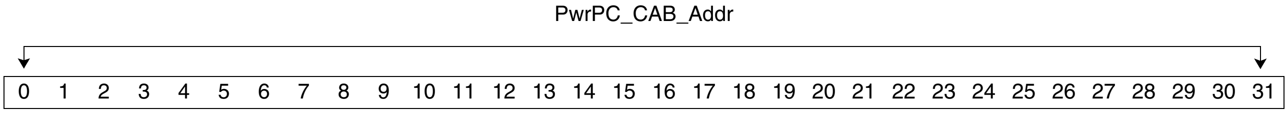

Start
Hardware/Software Synchronization
Read CAB_Status
rd_rdy = 1
Y
N
Read CAB_Data and discard results
CAB read or write
rd
wr
Read CAB_Data and discard results
Pending CAB Read CAB I/F access retried
Rd_Rdy = ‘1’
Read CAB_Data (sets rd_rdy to ‘0’)
Write CAB_Data with data value
Pending CAB Write CAB I/F access retried
Finish

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 subsystem access to NP2G 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 subsystem access to NP2G structures via the CAB Interface.

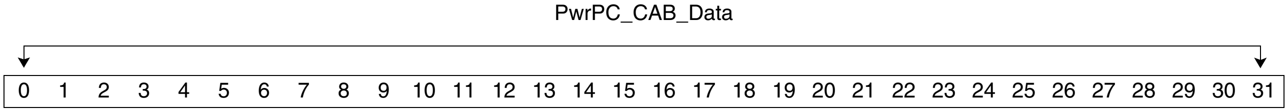
10.7.2 PowerPC CAB Data (PwrPC_CAB_Data) Register

The PowerPC CAB Data register is accessible from the PLB and contains the value of CAB data written or read when the PowerPC subsystem accesses NP2G 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'



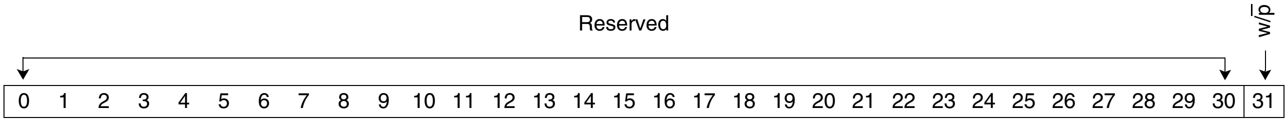
Field Name	Bit(s)	Reset	Description
PwrPC_CAB_Data	0:31	x'0000 0000'	CAB data written or read when the PowerPC subsystem accesses NP2G structures via the CAB Interface.

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 subsystem. The bit in this register indicates whether the wait or polled protocol is used when the PowerPC subsystem accesses CAB connected structures within the NP2G.

Access Type Read/Write

Base Address (PLB) x‘7800 0010’



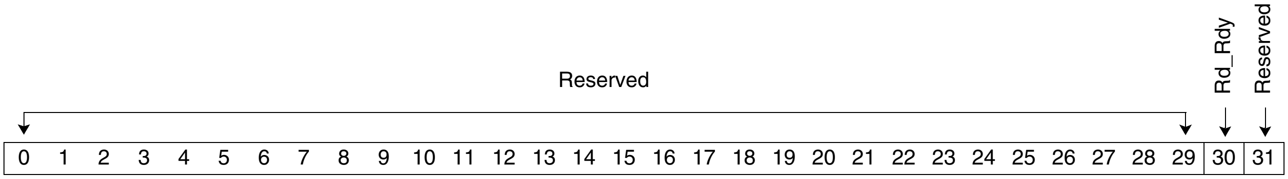
Field Name	Bit(s)	Reset	Description
Reserved	0:30		Reserved
w/$\overline{p}$	31	0	Wait or polled access control value 0 PowerPC subsystem polls the PwrPC_CAB_Status register to determine when access is complete 1 CAB Interface macro inserts wait states on the PLB until the CAB access is complete

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 subsystem accesses of CAB connected structures within the NP2G.

Access Type Read

Base Address (PLB) x‘7800 0018’



Field Name	Bit(s)	Description
Reserved	0:29	Reserved
Rd_Rdy	30	Read Data Ready indicator (used for polled access mode w/$\overline{p}$ = ‘0’ only) 0 No data in CAB Data register, a new CAB access can begin. 1 Data from a CAB read access is waiting in the CAB Data register
Reserved	31	Reserved

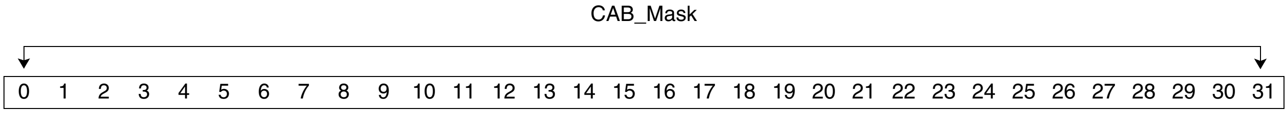
10.7.5 PowerPC CAB Mask (PwrPC_CAB_Mask) Register

The PowerPC CAB Mask (PwrPC_CAB_Mask) register is accessible from the PLB and supplies a mask value used in conjunction with a write-under-mask CAB access. Each ‘1’ bit of the mask indicates which bits of the CAB register will be updated. The corresponding bit value of the PwrPC_CAB_WUM_Data register is used to update the CAB register. All other bits of the CAB register will remain unaltered.

Hardware Reset x‘0000 0000’

PLB Address x‘7800 0020’ PowerPC CAB Mask register

Access Type Read/Write



Field Name	Bit(s)	Description
PwrPC_CAB_Mask	0:31	CAB mask value for PowerPC subsystem access to IBM Network Processor structures via the CAB Interface. Each ‘1’ bit of the mask indicates which bits of the CAB register will be updated.

10.7.6 PowerPC CAB Write Under Mask Data (PwrPC_CAB_WUM_Data)

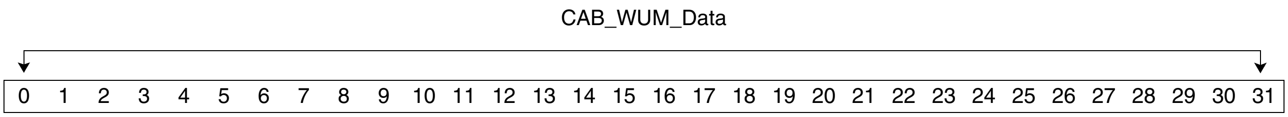
The PowerPC CAB Data (PwrPC_CAB_WUM_Data) register is accessible from the PLB and contains the value of CAB data written when the PowerPC subsystem accesses IBM Network Processor structures via the CAB Interface using the write-under-mask function. Writing to the PwrPC_CAB_WUM_Data register has the side effect of initiating a write-under-mask access on the CAB. The data value written to the PwrPC_CAB_WUM_Data register is combined with the contents of the PwrPC_CAB_Mask register and the contents of the CAB register indicated by the PwrPC_CAB_Addr register. For each bit location in which the value of the PwrPC_CAB_Mask register is ‘1’, the corresponding bit of the CAB register is updated with the contents of the PwrPC_CAB_WUM_Data register. All other bits of the CAB register remain unaltered.

The bits of this register are shared with the PwrPC_CAB_Data register. Writing to this register will alter the contents of the PwrPC_CAB_Data register.

Hardware Reset x‘0000 0000’

PLB Address x‘7800 0028’ PowerPC CAB WUM Data register

Access Type Write



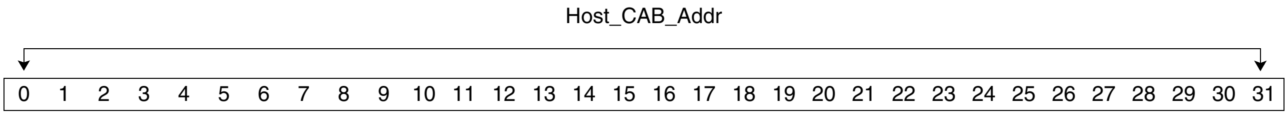
Field Name	Bit(s)	Description
PwrPC_CAB_WUM_Data	0:31	CAB data to be combined with the PwrPC_CAB_Mask register when the PowerPC subsystem accesses IBM Network Processor structures via the CAB Interface in write-under_mask mode.

10.7.7 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 NP2G 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 NP2G structures via the CAB Interface.

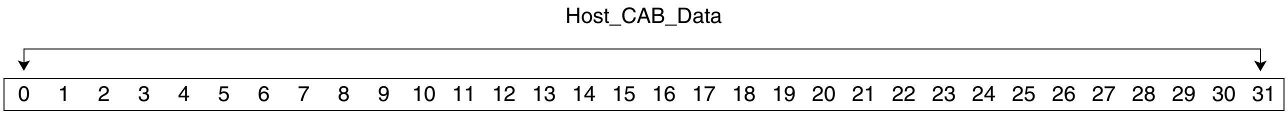
10.7.8 PCI Host CAB Data (Host_CAB_Data) Register

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 NP2G 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

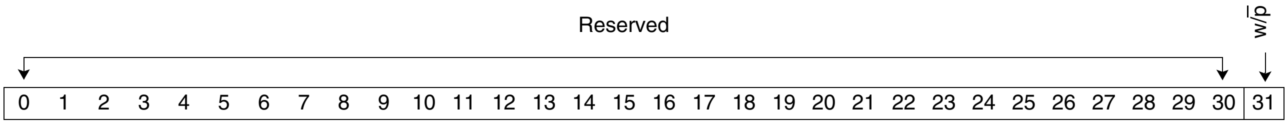
Base Address (PLB) x‘7800 8008’



Field Name	Bit(s)	Reset	Description
Host_CAB_Data	0:31	x’0000 0000’	CAB data written or read when the PCI Host accesses NP2G structures via the CAB Interface.

10.7.9 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 NP2G.



Access Type Read/Write

Base Address (PLB) x‘7800 8010’

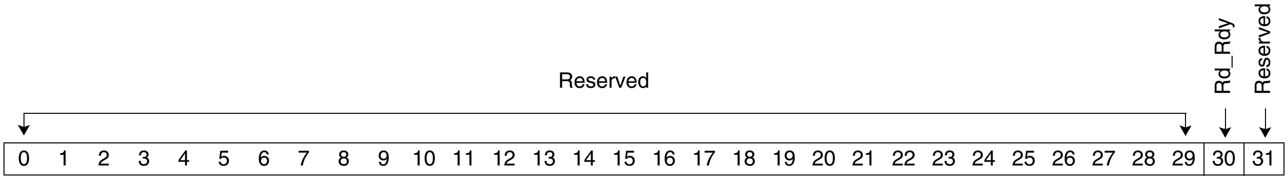
Field Name	Bit(s)	Reset	Description
Reserved	0:30		Reserved
w/$\overline{p}$	31	0	Wait or polled access control value 0 PCI Host polls the Host_CAB_Status register to determine when access is complete 1 CAB Interface macro inserts wait states on the PLB until the CAB access is complete

10.7.10 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 NP2G.

Access Type Read

Base Address (PLB) x‘7800 8018’



Field Name	Bit(s)	Description
Reserved	0:29	Reserved
Rd_Rdy	30	Read Data Ready indicator (used for polled access mode w/$\overline{p}$ = ‘0’ only) 0 No data in CAB Data register, a new CAB access can begin. 1 Data from a CAB read access is waiting in the CAB Data register
Reserved	31	Reserved

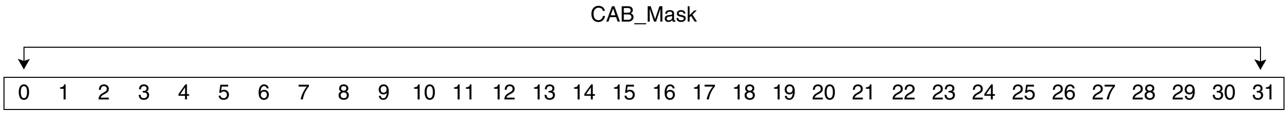
10.7.11 PCI Host CAB Mask (Host_CAB_Mask) Register

The PCI Host CAB Mask (PwrPC_CAB_Mask) register is accessible from the PLB and supplies a mask value used in conjunction with a write-under-mask CAB access. Each ‘1’ bit of the mask indicates which bits of the CAB register will be updated. The corresponding bit value of the Host_CAB_WUM_Data register is used to update the CAB register. All other bits of the CAB register will remain unaltered.

Hardware Reset x‘0000 0000’

PLB Address x‘7800 8020’ PCI Host CAB Mask register

Access Type Read/Write



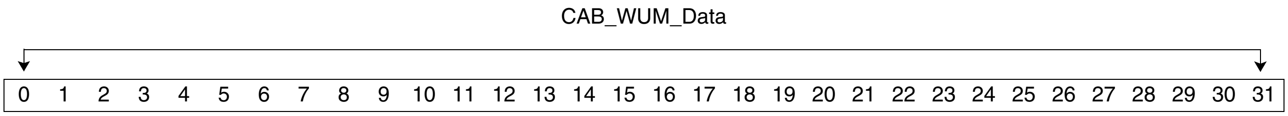
Field Name	Bit(s)	Description
Host_CAB_Mask	0:31	CAB mask value for PCI Host access to IBM Network Processor structures via the CAB Interface. Each ‘1’ bit of the mask indicates which bits of the CAB register will be updated.

10.7.12 PCI Host CAB Write Under Mask Data (Host_CAB_WUM_Data) Register

The PCI Host CAB Data (Host_CAB_WUM_Data) register is accessible from the PLB and contains the value of CAB data written when the PCI Host accesses IBM Network Processor structures via the CAB Interface using the write-under-mask function. Writing to the Host_CAB_WUM_Data register has the side effect of initiating a write-under-mask access on the CAB. The data value written to the PwrPC_CAB_WUM_Data register is combined with the contents of the PwrPC_CAB_Mask register and the contents of the CAB register indicated by the PwrPC_CAB_Addr register. For each bit location in which the value of the Host_CAB_Mask register is '1', the corresponding bit of the CAB register is updated with the contents of the Host_CAB_WUM_Data register. All other bits of the CAB register remain unaltered.

The bits of this register are shared with the Host_CAB_Data register. Writing to this register will alter the contents of the Host_CAB_Data register.

Hardware Reset	x'0000 0000'
PLB Address	x'7800 8028' PCI Host CAB WUM Data register
Access Type	Write



Field Name	Bit(s)	Description
Host_CAB_WUM_Data	0:31	CAB data to be combined with the Host_CAB_Mask register when the PCI Host accesses IBM Network Processor structures via the CAB Interface in write-under_mask mode.

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 NP2G'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 subsystem.

The Mailbox and DRAM Interface macro also provides redirection of boot code. This function is used in system implementations in which the NP2G 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 processor'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 Subsystem

Communication between the PCI Host and the PowerPC subsystem is accomplished through PCI Host to PowerPC subsystem (H2P) and PowerPC subsystem to PCI Host (P2H) interrupts. The P2H interrupt is implemented by asserting the NP2G'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 subsystem 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 subsystem by setting bit 13 of the PCI Macro's Bridge Options 2 register. This interrupt signal is applied to the PowerPC Universal Interrupt Controller (UIC).

Communications between the PCI Host and the PowerPC subsystem use message buffers in PCI address space. Software running in the PCI Host manages these buffers. For communications from the PowerPC subsystem 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 subsystem 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 subsystem 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 subsystem by writing its starting PCI address value into the H2P_Msg_Addr register. Writing this register sets the H2P_Msg_Interr to the PowerPC UIC and sets the H2P_Msg_Busy bit of the Msg_Status register. An interrupt is generated when enabled by the UIC. The PowerPC subsystem 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 subsystem has finished processing the data buffer, allowing it to be reused.

The P2H_Msg_Addr register is written by the PowerPC subsystem and read by the PCI Host processor. Whenever the PowerPC subsystem 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 subsystem and the resetting of the interrupt by the PCI Host (only the PowerPC subsystem writes this register and only the PCI Host reads this register).

The Doorbell register is written and read by either the PowerPC subsystem 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 subsystem or the PCI Host processor is writing the register.

When written by the PowerPC subsystem, 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.

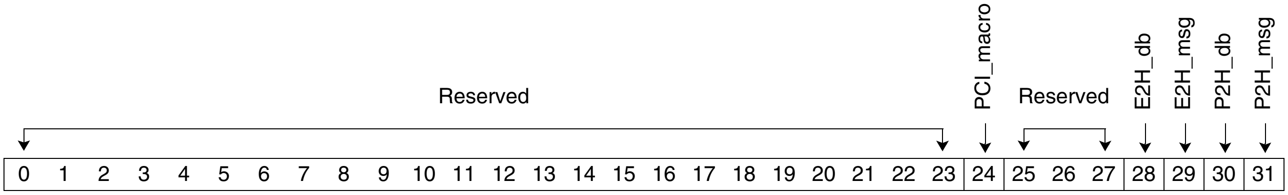
Any of the signals recorded as '1' in the PCI Interrupt Status register activates the INTA# signal if the corresponding condition is enabled in the NP2G's PCI Interrupt Enable register. The PCI Host processor reads the PCI Interrupt Status register to determine the interrupt source.

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 NP2G. 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'



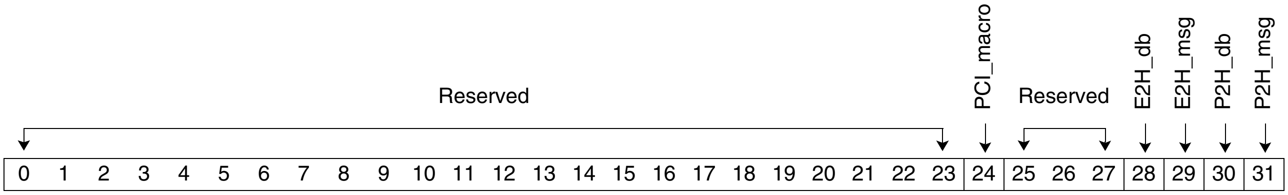
Field Name	Bit(s)	Reset	Description
Reserved	0:23		Reserved
PCI_macro	24	0	PCI interrupt from macro indicator. A PCI interrupt is asserted by the PLB to PCI macro when bit 0 of the macro's PCIICS register is set by software to a value of '1'. See the *IBM PowerPC 405GP Embedded Processor User's Manual* for details. 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 subsystem 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 subsystem 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 NP2G.

Access Type Read/Write

Base Address (PLB) x‘7801 0008’



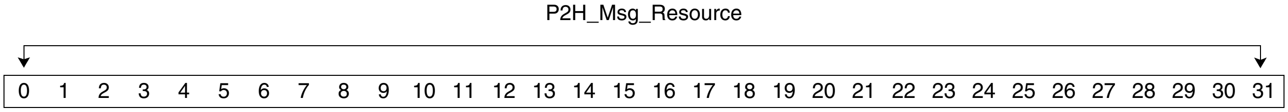
Field Name	Bit(s)	Reset	Description
Reserved	0:23		Reserved
PCI_macro	24	0	PCI macro interrupt - controls the assertion of a PCI interrupt from the PCI macro. A PCI interrupt is asserted by the PLB to PCI macro when bit 0 of the macro's PCIICS register is set by software to a value of '1'. See the *IBM PowerPC 405GP Embedded Processor User’s Manual* for details. 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 subsystem 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 subsystem 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 Subsystem to PCI Host Message Resource (P2H_Msg_Resource) Register

The PowerPC Subsystem to PCI Host Message Resource register is accessible from the PLB. The PowerPC subsystem uses this register to obtain message buffers in PCI address space for messages the PowerPC subsystem 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 found in the Message Status Register (see *10.8.23 Message Status (Msg_Status) Register* on page 404) and reading the P2H_Msg_Resource register clears this flag.

Access Type Read/Write

Base Address (PLB) x‘7801 0010’
(Additional actions occur when reading this register using this address. See register description for details.)



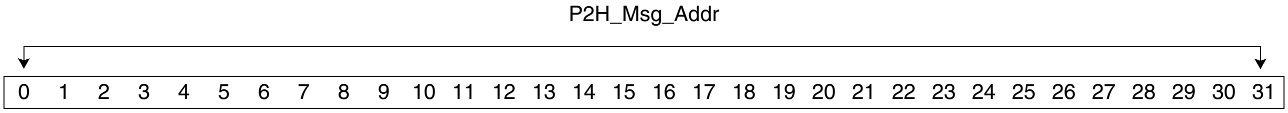
Field Name	Bit(s)	Reset	Description
P2H_Msg_Resource	0:31	x’0000 0000’	PowerPC subsystem to PCI Host Message Resource value, written with the PCI starting address of a message buffer. Writing this register sets to 1 the P2H_Bufr_Valid flag found in the Message Status Register (see *10.8.23 Message Status (Msg_Status) Register* on page 404). Reading this register sets to 0 the P2H_Bufr_Valid flag.

10.8.5 PowerPC Subsystem to Host Message Address (P2H_Msg_Addr) Register

The PowerPC Subsystem to PCI Host Message Address register is accessible from the PLB and is used by the PowerPC subsystem 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 subsystem to PCI Host Message address value, indicates the PCI starting address of a message.

10.8.6 PowerPC Subsystem to Host Doorbell (P2H_Doorbell) Register

The PowerPC Subsystem to PCI Host Doorbell register is accessible from the PLB and is used by the PowerPC subsystem to signal interrupts to the PCI Host processor. The PowerPC subsystem 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

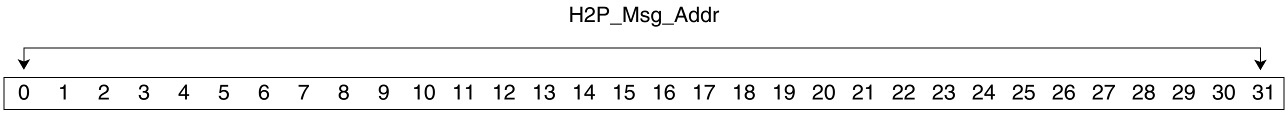
P2H_Msg_Doorbell 31	P2H_Msg_Doorbell 30	P2H_Msg_Doorbell 29	P2H_Msg_Doorbell 28	P2H_Msg_Doorbell 27	P2H_Msg_Doorbell 26	P2H_Msg_Doorbell 25	P2H_Msg_Doorbell 24	P2H_Msg_Doorbell 23	P2H_Msg_Doorbell 22	P2H_Msg_Doorbell 21	P2H_Msg_Doorbell 20	P2H_Msg_Doorbell 19	P2H_Msg_Doorbell 18	P2H_Msg_Doorbell 17	P2H_Msg_Doorbell 16	P2H_Msg_Doorbell 15	P2H_Msg_Doorbell 14	P2H_Msg_Doorbell 13	P2H_Msg_Doorbell 12	P2H_Msg_Doorbell 11	P2H_Msg_Doorbell 10	P2H_Msg_Doorbell 9	P2H_Msg_Doorbell 8	P2H_Msg_Doorbell 7	P2H_Msg_Doorbell 6	P2H_Msg_Doorbell 5	P2H_Msg_Doorbell 4	P2H_Msg_Doorbell 3	P2H_Msg_Doorbell 2	P2H_Msg_Doorbell 1	P2H_Msg_Doorbell 0
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Field Name	Bit(s)	Reset	Description
P2H_Msg_Doorbell 31	0	0	PowerPC subsystem 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 Subsystem Message Address (H2P_Msg_Addr) Register

The PCI Host to PowerPC Subsystem Message Address register is accessible from the PLB and is used by the PCI Host to send messages to the PowerPC subsystem’s 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 UIC. When this interrupt is enabled, an interrupt to the PowerPC subsystem is generated. Reading this register resets the H2P_Msg_Interr input to the UIC. Reading this register at the alternate PLB address resets the Message Status register’s H2P_Msg_Busy bit (see *10.8.23 Message Status (Msg_Status) Register* on page 404).

Access Type	Alternate	Read
	Primary	Read/Write
Base Address (PLB)	Alternate	x‘7801 0050’ (Additional actions occur when reading this register using this address. See register description for details.)
	Primary	x‘7801 0060’ (Additional actions occur when reading this register using this address. See register description for details.)



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 Subsystem Doorbell (H2P_Doorbell) Register

The PCI Host to PowerPC Subsystem Doorbell (H2P_Doorbell) register is accessible from the PLB and is used by the PCI Host processor to signal interrupts to the PowerPC subsystem. 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 UIC is activated. The PowerPC subsystem reads this register to determine which of the doorbells have been activated. The PowerPC subsystem 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

H2P_Msg_Doorbell 31	H2P_Msg_Doorbell 30	H2P_Msg_Doorbell 29	H2P_Msg_Doorbell 28	H2P_Msg_Doorbell 27	H2P_Msg_Doorbell 26	H2P_Msg_Doorbell 25	H2P_Msg_Doorbell 24	H2P_Msg_Doorbell 23	H2P_Msg_Doorbell 22	H2P_Msg_Doorbell 21	H2P_Msg_Doorbell 20	H2P_Msg_Doorbell 19	H2P_Msg_Doorbell 18	H2P_Msg_Doorbell 17	H2P_Msg_Doorbell 16	H2P_Msg_Doorbell 15	H2P_Msg_Doorbell 14	H2P_Msg_Doorbell 13	H2P_Msg_Doorbell 12	H2P_Msg_Doorbell 11	H2P_Msg_Doorbell 10	H2P_Msg_Doorbell 9	H2P_Msg_Doorbell 8	H2P_Msg_Doorbell 7	H2P_Msg_Doorbell 6	H2P_Msg_Doorbell 5	H2P_Msg_Doorbell 4	H2P_Msg_Doorbell 3	H2P_Msg_Doorbell 2	H2P_Msg_Doorbell 1	H2P_Msg_Doorbell 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Field Name	PLB Bit(s)	Reset	Description
H2P_Msg_Doorbell 31	0	0	PCI Host to PowerPC subsystem 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 Subsystem and EPC

Communication between the PowerPC subsystem and the EPC is accomplished by writing message data into buffers in the PowerPC DRAM (D6) and signalling the destination processor with an interrupt. The PowerPC software manages message data buffers for PowerPC Subsystem to EPC (P2E) messages and EPC to PowerPC Subsystem (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. 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 UIC. The PowerPC subsystem reads this register to find and process the message and, due to the read, the interrupt condition is cleared. To avoid message loss, the PowerPC should not replenish the E2P_Msg_Resource register with a new buffer address value until the EPC has returned the preceding buffer by writing the E2P_Msg_Addr register.

There is no need for a P2E_Msg_Resource register because the PowerPC software manages the message data buffers The PowerPC subsystem 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 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 subsystem will poll the busy flag to determine when the buffer can be reused.

10.8.10 EPC to PowerPC Subsystem Resource (E2P_Msg_Resource) Register

The PowerPC subsystem accesses the EPC to PowerPC Subsystem 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 DRAM address space for messages it sends to the PowerPC processor. The PowerPC processor 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 (see *10.8.23 Message Status (Msg_Status) Register* on page 404). Reading the E2P_Msg_Resource register from the CAB resets this flag.

Access Type	CAB	Read (See above description for additional actions that occur during a read of this register using this address.)
	PLB	Read/Write (See above description for additional actions that occur during a read of this register using this address.)
Base Address	CAB	x‘3801 0010’
	PLB	x‘7801 0040’

CAB View

E2P_Msg_Resource

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

PLB View

E2P_Msg_Resource

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

CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Resource	31:0	x’0000 0000’	EPC to PowerPC Subsystem Message Resource - written with the PowerPC 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 Subsystem Message Resource - written with the PowerPC DRAM starting address of a message buffer.

10.8.11 EPC to PowerPC Subsystem Message Address (E2P_Msg_Addr) Register

The PowerPC subsystem accesses the EPC to PowerPC Subsystem 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 DRAM address at which the message begins. Writing to this register sets E2P_Msg_Interr input to the PowerPC UIC. When the UIC is configured to enable this input, an interrupt signal to the PowerPC processor is activated. Reading the E2P_Msg_Addr register via the PLB address resets the E2P_Msg_Interr input to the PowerPC UIC.

Access Type	CAB	Read/Write
	PLB	Read
Base Address	CAB	x‘3801 0020’
	PLB	x‘7801 0048’ (Additional actions occur when reading this register using this address. See register description for details.)

CAB View

E2P_Msg_Addr

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

PLB View

E2P_Msg_Addr

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

CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Addr	0:31	x’0000 0000’	EPC to PowerPC Subsystem 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 Subsystem Message Address - indicates the PCI starting address of a message.

10.8.12 EPC to PowerPC Subsystem Doorbell (E2P_Doorbell) Register

The PowerPC subsystem accesses the EPC to PowerPC Subsystem 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 subsystem. 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 UIC is activated. The PowerPC subsystem reads this register to determine which of the doorbells have been activated. The PowerPC subsystem 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

E2P_Doorbells

E2P_Msg_Doorbell 31	E2P_Msg_Doorbell 30	E2P_Msg_Doorbell 29	E2P_Msg_Doorbell 28	E2P_Msg_Doorbell 27	E2P_Msg_Doorbell 26	E2P_Msg_Doorbell 25	E2P_Msg_Doorbell 24	E2P_Msg_Doorbell 23	E2P_Msg_Doorbell 22	E2P_Msg_Doorbell 21	E2P_Msg_Doorbell 20	E2P_Msg_Doorbell 19	E2P_Msg_Doorbell 18	E2P_Msg_Doorbell 17	E2P_Msg_Doorbell 16	E2P_Msg_Doorbell 15	E2P_Msg_Doorbell 14	E2P_Msg_Doorbell 13	E2P_Msg_Doorbell 12	E2P_Msg_Doorbell 11	E2P_Msg_Doorbell 10	E2P_Msg_Doorbell 9	E2P_Msg_Doorbell 8	E2P_Msg_Doorbell 7	E2P_Msg_Doorbell 6	E2P_Msg_Doorbell 5	E2P_Msg_Doorbell 4	E2P_Msg_Doorbell 3	E2P_Msg_Doorbell 2	E2P_Msg_Doorbell 1	E2P_Msg_Doorbell 0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PLB View

E2P_Doorbells

E2P_Msg_Doorbell 31	E2P_Msg_Doorbell 30	E2P_Msg_Doorbell 29	E2P_Msg_Doorbell 28	E2P_Msg_Doorbell 27	E2P_Msg_Doorbell 26	E2P_Msg_Doorbell 25	E2P_Msg_Doorbell 24	E2P_Msg_Doorbell 23	E2P_Msg_Doorbell 22	E2P_Msg_Doorbell 21	E2P_Msg_Doorbell 20	E2P_Msg_Doorbell 19	E2P_Msg_Doorbell 18	E2P_Msg_Doorbell 17	E2P_Msg_Doorbell 16	E2P_Msg_Doorbell 15	E2P_Msg_Doorbell 14	E2P_Msg_Doorbell 13	E2P_Msg_Doorbell 12	E2P_Msg_Doorbell 11	E2P_Msg_Doorbell 10	E2P_Msg_Doorbell 9	E2P_Msg_Doorbell 8	E2P_Msg_Doorbell 7	E2P_Msg_Doorbell 6	E2P_Msg_Doorbell 5	E2P_Msg_Doorbell 4	E2P_Msg_Doorbell 3	E2P_Msg_Doorbell 2	E2P_Msg_Doorbell 1	E2P_Msg_Doorbell 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

CAB View

Field Name	Bit(s)	Reset	Description
E2P_Msg_Doorbell 31	31	0	EPC to PowerPC Subsystem 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 Subsystem 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 NP2G. This register’s bits are set by hardware and are read by EPC software to determine the source of interrupts.

10.8.14 EPC Interrupt Mask Register

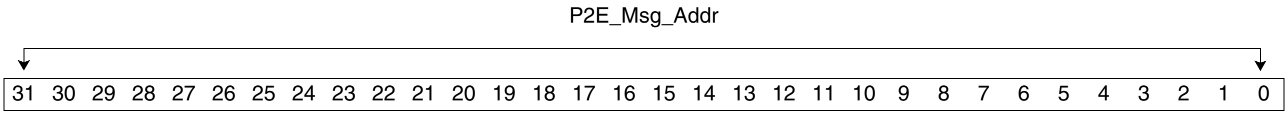
The EPC contains a Interrupt Mask 2 register which is accessible from the CAB Interface. This register enables EPC interrupts from sources within the NP2G.

10.8.15 PowerPC Subsystem to EPC Message Address (P2E_Msg_Addr) Register

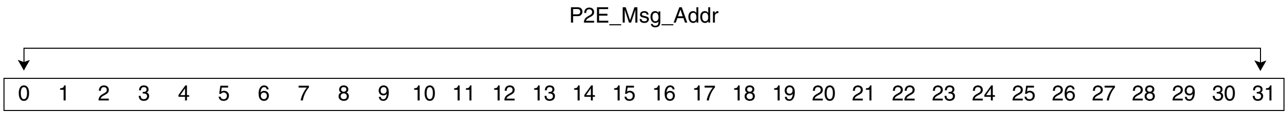
The PowerPC subsystem accesses the PowerPC Subsystem to EPC Message Address register from the PLB, while the EPC accesses this register from the CAB Interface. This register is used by the PowerPC subsystem to send messages to the EPC. The value written into this register is the PowerPC DRAM address 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 (see *10.8.23 Message Status (Msg_Status) Register* on page 404).

Access Type	Primary	Read
	Alternate	Read (Additional actions occur when reading this register using this address. See register description for details.)
	PLB	Read/Write
Base Address	Primary	x'3801 0080'
	Alternate	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 Subsystem 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 Subsystem to EPC Message Address - indicates a message's PowerPC DRAM starting address.

10.8.16 PowerPC Subsystem to EPC Doorbell (P2E_Doorbell) Register

The PowerPC subsystem accesses the PowerPC Subsystem to EPC Doorbell register from the PLB, while the EPC accesses this register from the CAB Interface. The PowerPC subsystem uses this register to signal interrupts to the EPC. The PowerPC subsystem 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 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

P2E_Doorbells

P2E_Msg_Doorbell 31	P2E_Msg_Doorbell 30	P2E_Msg_Doorbell 29	P2E_Msg_Doorbell 28	P2E_Msg_Doorbell 27	P2E_Msg_Doorbell 26	P2E_Msg_Doorbell 25	P2E_Msg_Doorbell 24	P2E_Msg_Doorbell 23	P2E_Msg_Doorbell 22	P2E_Msg_Doorbell 21	P2E_Msg_Doorbell 20	P2E_Msg_Doorbell 19	P2E_Msg_Doorbell 18	P2E_Msg_Doorbell 17	P2E_Msg_Doorbell 16	P2E_Msg_Doorbell 15	P2E_Msg_Doorbell 14	P2E_Msg_Doorbell 13	P2E_Msg_Doorbell 12	P2E_Msg_Doorbell 11	P2E_Msg_Doorbell 10	P2E_Msg_Doorbell 9	P2E_Msg_Doorbell 8	P2E_Msg_Doorbell 7	P2E_Msg_Doorbell 6	P2E_Msg_Doorbell 5	P2E_Msg_Doorbell 4	P2E_Msg_Doorbell 3	P2E_Msg_Doorbell 2	P2E_Msg_Doorbell 1	P2E_Msg_Doorbell 0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PLB View

P2E_Doorbells

P2E_Msg_Doorbell 31	P2E_Msg_Doorbell 30	P2E_Msg_Doorbell 29	P2E_Msg_Doorbell 28	P2E_Msg_Doorbell 27	P2E_Msg_Doorbell 26	P2E_Msg_Doorbell 25	P2E_Msg_Doorbell 24	P2E_Msg_Doorbell 23	P2E_Msg_Doorbell 22	P2E_Msg_Doorbell 21	P2E_Msg_Doorbell 20	P2E_Msg_Doorbell 19	P2E_Msg_Doorbell 18	P2E_Msg_Doorbell 17	P2E_Msg_Doorbell 16	P2E_Msg_Doorbell 15	P2E_Msg_Doorbell 14	P2E_Msg_Doorbell 13	P2E_Msg_Doorbell 12	P2E_Msg_Doorbell 11	P2E_Msg_Doorbell 10	P2E_Msg_Doorbell 9	P2E_Msg_Doorbell 8	P2E_Msg_Doorbell 7	P2E_Msg_Doorbell 6	P2E_Msg_Doorbell 5	P2E_Msg_Doorbell 4	P2E_Msg_Doorbell 3	P2E_Msg_Doorbell 2	P2E_Msg_Doorbell 1	P2E_Msg_Doorbell 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

CAB View

Field Name	Bit(s)	Reset	Description
P2E_Msg_Doorbell 31	31	0	PowerPC Subsystem 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 Subsystem 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 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 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 PCI Host processor 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 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 found in the Message Status register (see *10.8.23 Message Status (Msg_Status) Register* on page 404). Reading the E2H_Msg_Resource register from the CAB resets this flag.

Access Type	CAB	Read (Additional actions occur when reading this register using this address. See register description for details.)
	PLB	Read/Write
Base Address	CAB	x‘3801 0200’
	PLB	x‘7801 0080’

CAB View

E2H_Msg_Resource

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

PLB View

E2H_Msg_Resource

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

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 DRAM starting address of a message buffer. When reading this register via the CAB, the E2H_Bufr_Valid flag found in the Message Status Register (see *10.8.23 Message Status (Msg_Status) Register* on page 404) is set to 0.

PLB View

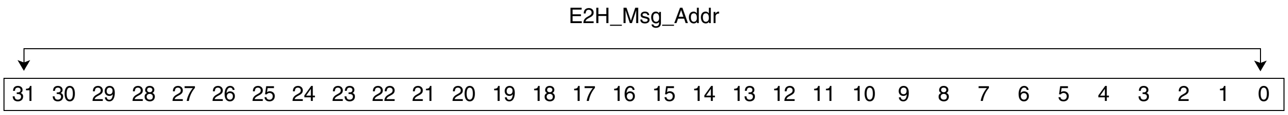
Field Name	Bit(s)	Reset	Description
E2H_Msg_Resource	0:31	x’0000 0000’	EPC to PCI Host Message Resource - written with the PowerPC DRAM starting address of a message buffer. When writing this register, the E2H_Bufr_Valid flag is set to 1.

10.8.19 EPC to PCI Host Message Address (E2H_Msg_Addr) Register

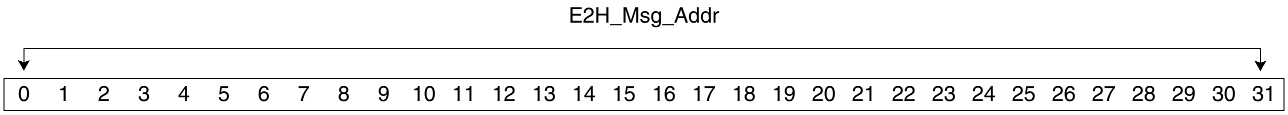
The PCI Host Processor accesses the EPC to PCI Host Message Address register from the PLB while the EPC accesses this register from its CAB Interface. The EPC uses this register to send messages to the PCI Host processor. The value written into this register is the PowerPC DRAM address at which the message 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’ (Additional actions occur when reading this register using this address. See register description for details.)

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 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 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

E2H_Doorbells

E2H_Msg_Doorbell 31	E2H_Msg_Doorbell 30	E2H_Msg_Doorbell 29	E2H_Msg_Doorbell 28	E2H_Msg_Doorbell 27	E2H_Msg_Doorbell 26	E2H_Msg_Doorbell 25	E2H_Msg_Doorbell 24	E2H_Msg_Doorbell 23	E2H_Msg_Doorbell 22	E2H_Msg_Doorbell 21	E2H_Msg_Doorbell 20	E2H_Msg_Doorbell 19	E2H_Msg_Doorbell 18	E2H_Msg_Doorbell 17	E2H_Msg_Doorbell 16	E2H_Msg_Doorbell 15	E2H_Msg_Doorbell 14	E2H_Msg_Doorbell 13	E2H_Msg_Doorbell 12	E2H_Msg_Doorbell 11	E2H_Msg_Doorbell 10	E2H_Msg_Doorbell 9	E2H_Msg_Doorbell 8	E2H_Msg_Doorbell 7	E2H_Msg_Doorbell 6	E2H_Msg_Doorbell 5	E2H_Msg_Doorbell 4	E2H_Msg_Doorbell 3	E2H_Msg_Doorbell 2	E2H_Msg_Doorbell 1	E2H_Msg_Doorbell 0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PLB View

E2H_Doorbells

E2H_Msg_Doorbell 31	E2H_Msg_Doorbell 30	E2H_Msg_Doorbell 29	E2H_Msg_Doorbell 28	E2H_Msg_Doorbell 27	E2H_Msg_Doorbell 26	E2H_Msg_Doorbell 25	E2H_Msg_Doorbell 24	E2H_Msg_Doorbell 23	E2H_Msg_Doorbell 22	E2H_Msg_Doorbell 21	E2H_Msg_Doorbell 20	E2H_Msg_Doorbell 19	E2H_Msg_Doorbell 18	E2H_Msg_Doorbell 17	E2H_Msg_Doorbell 16	E2H_Msg_Doorbell 15	E2H_Msg_Doorbell 14	E2H_Msg_Doorbell 13	E2H_Msg_Doorbell 12	E2H_Msg_Doorbell 11	E2H_Msg_Doorbell 10	E2H_Msg_Doorbell 9	E2H_Msg_Doorbell 8	E2H_Msg_Doorbell 7	E2H_Msg_Doorbell 6	E2H_Msg_Doorbell 5	E2H_Msg_Doorbell 4	E2H_Msg_Doorbell 3	E2H_Msg_Doorbell 2	E2H_Msg_Doorbell 1	E2H_Msg_Doorbell 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

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 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 primary CAB address will reset the H2E_Msg_Interr signal to the EPC. Reading the H2E_Msg_Addr from the alternate CAB address will reset the H2E_Msg_Busy bit of the Msg_Status register (see *10.8.23 Message Status (Msg_Status) Register* on page 404).

Access Type	Primary	Read
	Alternate	Read
	PLB	Read/Write
Base Address	Primary	x'3801 1000' (Additional actions occur when reading this register using this address. See register description for details.)
	Alternate	x'3802 0020' (Additional actions occur when reading this register using this address. See register description for details.)
	PLB	x'7801 00A8'

CAB View

H2E_Msg_Addr																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PLB View

H2E_Msg_Addr																															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

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

H2E_Doorbells																															
H2E_Msg_Doorbell 31	H2E_Msg_Doorbell 30	H2E_Msg_Doorbell 29	H2E_Msg_Doorbell 28	H2E_Msg_Doorbell 27	H2E_Msg_Doorbell 26	H2E_Msg_Doorbell 25	H2E_Msg_Doorbell 24	H2E_Msg_Doorbell 23	H2E_Msg_Doorbell 22	H2E_Msg_Doorbell 21	H2E_Msg_Doorbell 20	H2E_Msg_Doorbell 19	H2E_Msg_Doorbell 18	H2E_Msg_Doorbell 17	H2E_Msg_Doorbell 16	H2E_Msg_Doorbell 15	H2E_Msg_Doorbell 14	H2E_Msg_Doorbell 13	H2E_Msg_Doorbell 12	H2E_Msg_Doorbell 11	H2E_Msg_Doorbell 10	H2E_Msg_Doorbell 9	H2E_Msg_Doorbell 8	H2E_Msg_Doorbell 7	H2E_Msg_Doorbell 6	H2E_Msg_Doorbell 5	H2E_Msg_Doorbell 4	H2E_Msg_Doorbell 3	H2E_Msg_Doorbell 2	H2E_Msg_Doorbell 1	H2E_Msg_Doorbell 0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PLB View

H2E_Doorbells																															
H2E_Msg_Doorbell 31	H2E_Msg_Doorbell 30	H2E_Msg_Doorbell 29	H2E_Msg_Doorbell 28	H2E_Msg_Doorbell 27	H2E_Msg_Doorbell 26	H2E_Msg_Doorbell 25	H2E_Msg_Doorbell 24	H2E_Msg_Doorbell 23	H2E_Msg_Doorbell 22	H2E_Msg_Doorbell 21	H2E_Msg_Doorbell 20	H2E_Msg_Doorbell 19	H2E_Msg_Doorbell 18	H2E_Msg_Doorbell 17	H2E_Msg_Doorbell 16	H2E_Msg_Doorbell 15	H2E_Msg_Doorbell 14	H2E_Msg_Doorbell 13	H2E_Msg_Doorbell 12	H2E_Msg_Doorbell 11	H2E_Msg_Doorbell 10	H2E_Msg_Doorbell 9	H2E_Msg_Doorbell 8	H2E_Msg_Doorbell 7	H2E_Msg_Doorbell 6	H2E_Msg_Doorbell 5	H2E_Msg_Doorbell 4	H2E_Msg_Doorbell 3	H2E_Msg_Doorbell 2	H2E_Msg_Doorbell 1	H2E_Msg_Doorbell 0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

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

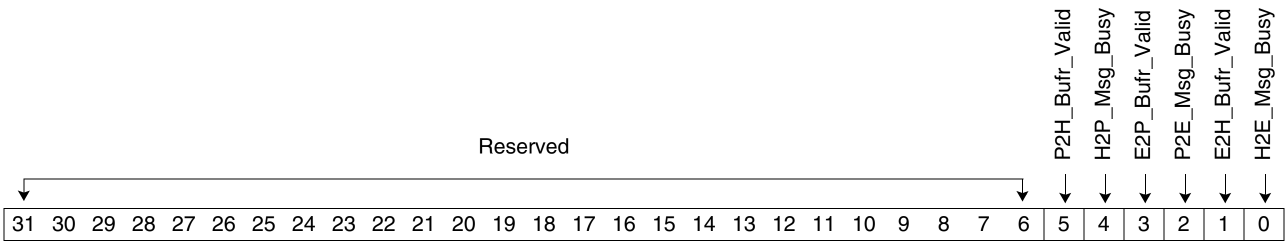
Field Name	PLB Bit	Reset	Description
H2E_Msg_Doorbell 31	0	0	PCI Host to EPC Doorbell - indicates which of the 32 possible doorbells is activated. 0 Not activated 1 Activated
H2E_Msg_Doorbell 30:1	1:30	0 for all	
H2E_Msg_Doorbell 0	31	0	

10.8.23 Message Status (Msg_Status) Register

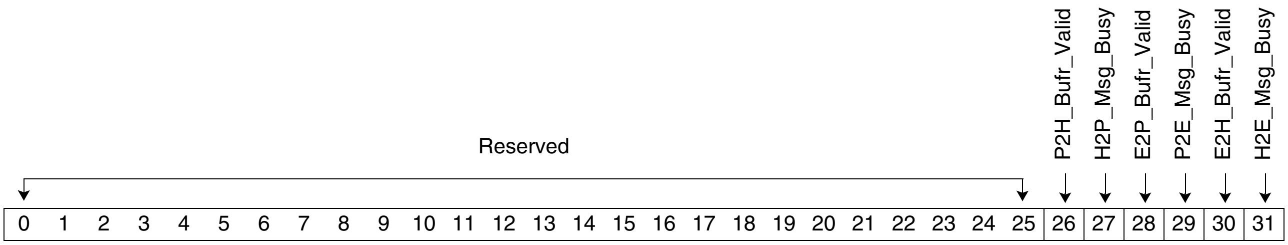
The Message Status register provides status information associated with inter-processor messaging. This read only register is accessible from either the PLB or the CAB for the purpose of checking status associated with messaging.

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 subsystem to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2P_Msg_Busy	4	0	PCI Host to PowerPC subsystem message busy indicator. 0 Not busy processing H2P message 1 Busy processing H2P message
E2P_Bufr_Valid	3	0	EPC to PowerPC subsystem message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
P2E_Msg_Busy	2	0	PowerPC to EPC subsystem 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 subsystem to PCI Host message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
H2P_Msg_Busy	27	0	PCI Host to PowerPC subsystem message busy indicator. 0 Not busy processing H2P message 1 Busy processing H2P message
E2P_Bufr_Valid	28	0	EPC to PowerPC subsystem message buffer valid indicator. 0 Buffer not valid 1 Buffer valid
P2E_Msg_Busy	29	0	PowerPC subsystem 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 PowerPC Boot Redirection Instruction Registers (Boot_Redir_Inst)

In system implementations in which the embedded PowerPC subsystem 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 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'FFFF FFE0' - x'FFFF FFFC' These instructions redirect the PowerPC subsystem to boot from a location in the PowerPC DRAM and are configured before the PPC_Reset is cleared.

Access Type Read/Write

Base Address x'3800 0110' - x'3800 0117'

Boot_Redir_Inst

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

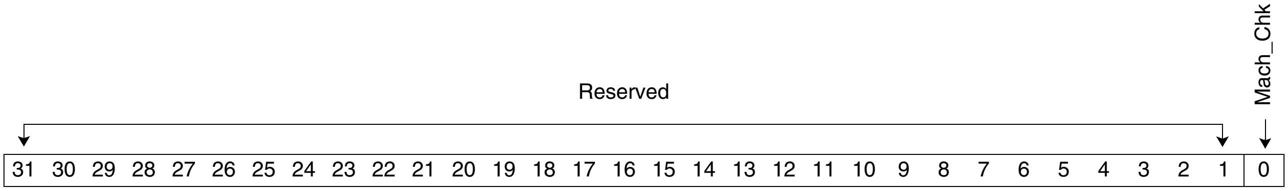
Field Name	Bit(s)	Reset	Description
Boot_Redir_Inst	31:0	x'0000 0000'	PowerPC boot redirection instruction address values contains instructions used by the PowerPC subsystem to redirect the boot sequence to a location in D6 DRAM. Offset Corresponding PowerPC Instruction Address 0 x'FFFF FFE0' 1 x'FFFF 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'

10.8.25 PowerPC Machine Check (PwrPC_Mach_Chk) Register

This register is accessible on the CAB and indicates the status of the machine check output of the PowerPC processor core.

Access Type Read

Base Address (CAB) x‘3800 0210’



Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved
Mach_Chk	0	0	PowerPC machine check indication. 0 No machine check has occurred 1 A machine check has occurred

10.8.26 Parity Error Status and Reporting

When reading the PowerPC subsystem memory, any parity errors encountered are recorded and actions are taken. PowerPC subsystem memory parity errors are indicated for each double word returned on the PLB regardless of the object size read. For this reason, parity must be initialized for every double word in each memory region used. Furthermore, for PowerPC processor code regions, correct parity must be initialized for each cache line in the region. Parity will be set correctly when the memory is written. Therefore, for code regions, this parity initialization is accomplished automatically when the code is loaded. Likewise, this initialization is performed if a diagnostic is performed on the memory that writes to every location. In all other cases, this parity initialization must be performed for each region of memory used prior to use.

Since even bytes and odd bytes of the PowerPC subsystem memory are stored in physically distinct modules, error statistics are collected on an odd and even byte basis. This information may prove useful in isolation of a failing part. The word address value of the most recent parity error is stored in the Slave Error Address Register (SEAR). In addition, bits of the Slave Error Status Register (SESR) indicate which PLB masters have experienced parity errors in reading the PowerPC subsystem memory. This status information is indicated by odd and even byte for each of the three PLB masters (Instruction Cache Unit, Data Cache Unit, and PCI/PLB macro).

Bits of the SESR are used to generate an interrupt input to the UIC. If any of these bits are set to b’1’ then the parity error interrupt input to the UIC is set to b’1’. When the UIC is configured to enable this interrupt input, an interrupt signal is generated to the PPC405 Core for processing. As a part of the interrupt service routine, the SESR must be read to clear the source of the interrupt.

10.8.27 Slave Error Address Register (SEAR)

The Slave Error Address Register records the word address value of the last occurrence of a parity error encountered by the DRAM Interface slave unit. This address value will isolate the location of the parity error to within a word.

Access Type Read

Base Address (PLB) x'7801 00B8'

Error Address

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

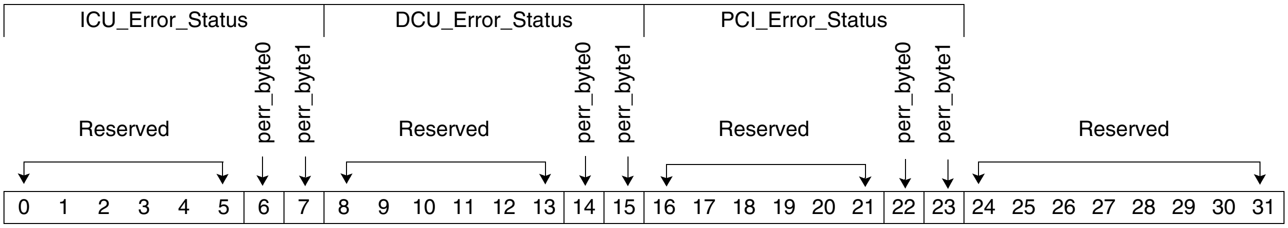
Field Name	Bit(s)	Reset	Description
Error Address	0:31	x'0000 0000'	Last PLB word address value that resulted in a parity error when using the DRAM Interface slave unit.

10.8.28 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’



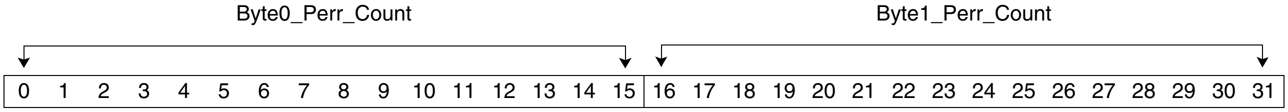
Field Name	Bit(s)	Reset	Description
Reserved	0:5		Reserved
Perr_Byte0	6	0	Byte 0 parity error indication - whether or not the Instruction Cache Unit PLB master encountered a parity error since the register was last read. 0 No parity error encountered 1 Parity error encountered
Perr_Byte1	7	0	Byte 1 parity 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

10.8.29 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 roll over 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 NP2G Resets

Table 10-10. Reset Domains

Reset Domain	Applies
PowerPC Core	Applies only to the PPC405 processor core and is used to control its start-up during system initialization
Network Function	Applies to all other functions in the NP2G

A general reset is performed whenever the NP2G's Reset input pin is activated. A general reset includes the PowerPC Core, and the NP2G Network Function reset domains. The PowerPC Core and the NP2G 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 NP2G'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 NP2G Network Function are in an idle condition with state machines active and control structures un-initialized. The NP2G Network Function will be functional once EPC picocode has been loaded and control structures are initialized. The EPC is activated automatically when loading boot picocode from the SPM interface or by setting the BD_Override bit to '1' in the Boot Override register (see *13.14.4 Boot Override Register (Boot_Override)* on page 463). The PowerPC Core is activated by clearing the bit of the PowerPC_Reset register.

The PowerPC Core and the NP2G 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 subsystem 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 PPC405 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' command is performed from RISCWatch and the Watch Dog Reset Enable is set to '1'. 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. The host processor (Boot_PPC = '0') loads code for the PowerPC processor and the EPC picoprocessor.

The general sequence of the start-up and initialization is as follows:

1. The host processor boots and performs board level configuration and testing while holding NP2G subsystems in reset.
2. The host processor releases the NP2G's reset signal.
3. Release of the general reset input pin of the NP2G activates its state machines. The NP2G is in an idle state with un-initialized structures and the PPC405 Core remains in reset and Host PCI Configuration is enabled.
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 NP2G'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 NP2G's DRAMs. This code includes the branch code loaded into the Boot_Redir_Inst registers.
7. The host loads the EPC instruction memory. This memory is 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 PPC405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface. The PowerPC subsystem 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 DRAM memory.
9. The host processor uses the CAB Interface to set the BD_Override bit to '1' in the Boot Override register (see *13.14.4 Boot Override Register (Boot_Override)* on page 463) to start the EPC code. This code controls the initialization of the NP2G 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 PPC405 Core.

10.9.3 Systems with PCI Host Processors and Initialized by PowerPC Subsystem

The PowerPC subsystem 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 (Boot_Picocode = '1'). The PowerPC subsystem boots from PCI address space (Boot_PPC = '1') and loads code for the PowerPC processor and the EPC picoprocessors.

The general sequence of the start-up and initialization is as follows:

1. The host processor boots and performs board level configuration and testing while holding NP2G subsystems in reset.
2. The host processor releases the NP2G's reset signal.
3. Release of the general reset input pin of the NP2G activates its state machines. The NP2G is in an idle state with un-initialized structures and the PPC405 Core remains in reset and Host PCI Configuration is enabled.
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 PPC405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
6. The PPC405 Core will boot from code residing in the PCI address space starting at address x'FFFF FFFC'.
7. The PPC405 Core processor configures and initializes the NP2G's DRAM controllers via the CAB Interface.
8. The PPC405 Core processor loads the PowerPC code, via the DRAM Interface Macro, into one of the NP2G's DRAMs.
9. The PPC405 Core processor loads the picocode for the EPC via the CAB Interface.
10. Using the CAB Interface, the host processor sets the BD_Override bit to '1' in the Boot Override register (see *13.14.4 Boot Override Register (Boot_Override)* on page 463) to start the EPC picocode. This picocode will control the initialization of the NP2G structures in preparation for network traffic. Alternatively, this initialization is performed by the PPC405 Core processor via the CAB Interface.
11. Communication ports are configured and enabled by the PPC405 Core.

10.9.4 Systems Without PCI Host Processors and Initialized by PowerPC Subsystem

The EPC initially controls start-up and initialization sequences of systems of this category, but they are primarily controlled by the PowerPC subsystem. These systems use the SPM Interface and its associated flash memory for booting the EPC picocode (Boot_Picocode = '0'). The PowerPC subsystem boots from PCI address space (Boot_PPC = '1') and loads code for the PowerPC processor and the EPC picoprocessor.

The general sequence of the start-up and initialization is as follows:

1. Release of the general reset input pin of the NP2G 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. The SPM Interface automatically starts the EPC and the PPC405 Core remains in reset.
2. EPC code executes diagnostic and initialization code which includes the initialization of the NP2G’s DRAM controllers.
3. The EPC starts the PPC405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
4. The PPC405 Core will boot from code residing in the PCI address space starting at address x‘FFFF FFFC’.
5. The PPC405 Core processor loads the PowerPC code via the DRAM Interface Macro into one of the NP2G’s DRAMs.
6. Communication ports are configured and enabled by the PPC405 Core.

10.9.5 Systems Without PCI Host or Delayed PCI Configuration 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 (Boot_Picocode = '0'). Code for the PowerPC subsystem and the EPC are loaded by the SPM Interface and the EPC (Boot_PPC = '0'). Code for the PowerPC subsystem 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 NP2G activates its state machines. EPC picocode is loaded from the flash memory via the SPM Interface. The SPM Interface automatically starts the EPC. The PPC405 Core remains in reset and PCI Host Configuration is disabled.
2. EPC code executes diagnostic and initialization code, which includes the initialization of the NP2G’s DRAM controllers.
3. The EPC loads the code for the PowerPC processor 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 PPC405 Core by clearing the PowerPC_Reset configuration register. This register is accessed via the CAB Interface.
5. The PPC405 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 DRAM memory.
6. If desired, the PPC405 Core can change the contents of the PCI Configuration Header registers and enable PCI Host Configuration.
7. Communication ports are configured and enabled by the PPC405 Core or, alternatively, by the EPC.

11. Reset and Initialization

This section provides, by example, a method for initializing the NP2G. It is not intended to be exhaustive in scope, since there are many configurations and environments where the NP2G 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 and is not sold separately.

11.1 Overview

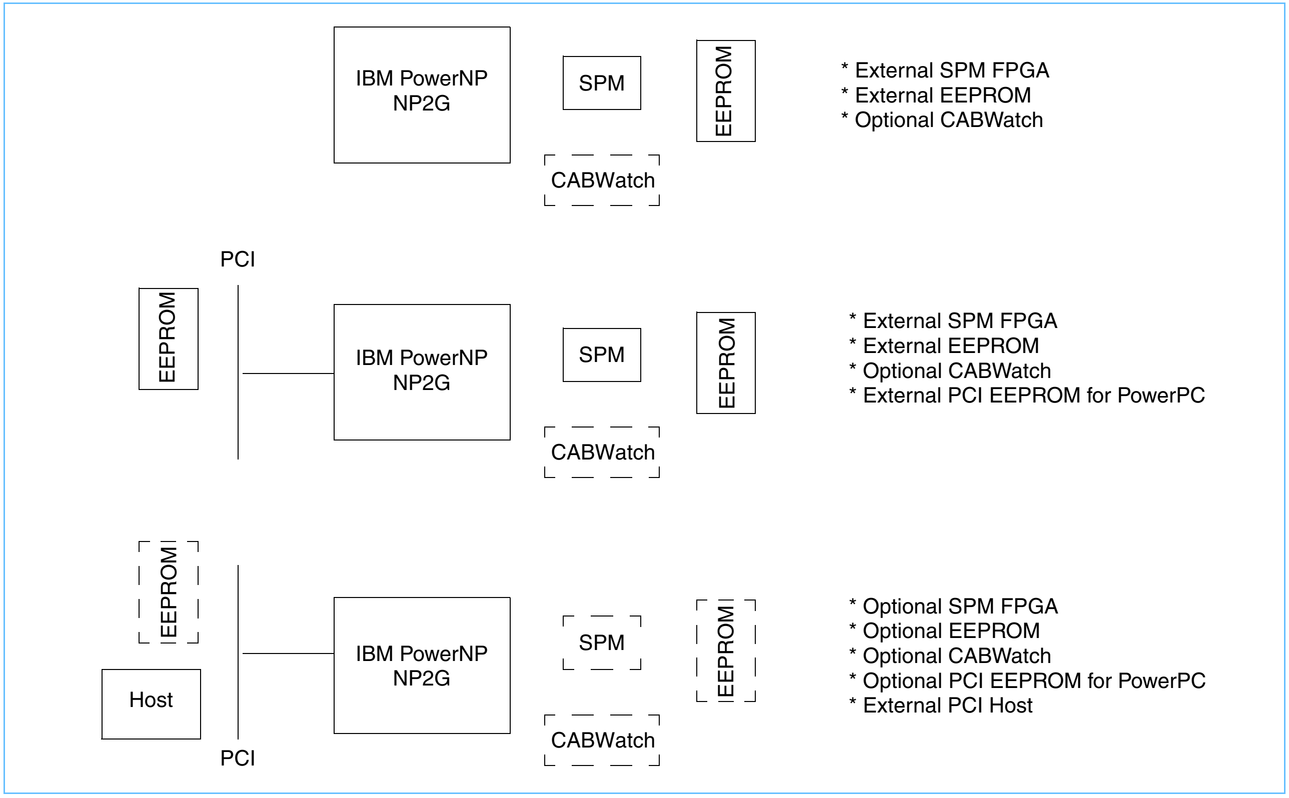
The NP2G supports a variety of system and adapter configurations. In order to support a particular environment, the network processor must be initialized with parameters that match that environment's system or adapter requirements. The sequence of reset and initialization is shown in *Table 11-1*.

Table 11-1. Reset and Initialization Sequence

Step	Action	Notes
1	Set I/Os	Device 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 NP2G supports the system environments shown in *Figure 11-1* on page 418 for Reset and Initialization.

Figure 11-1. 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

Several NP2G I/Os must be set to appropriate values 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 11-2. Set I/Os Checklist

I/O	Values	Notes
Testmode(1:0)	See *Table 2-31: Miscellaneous Pins* on page 75 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 2-31: Miscellaneous Pins* on page 75 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 2-31: Miscellaneous Pins* on page 75 for the encoding of this I/O.	Controls from where the internal PPC fetches its initial boot code.
PCI_Speed	See *Table 2-27: PCI Pins* on page 71 for the encoding of this I/O.	Controls the speed of the PCI bus.
CPDetect_A	See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for the encoding of this I/O.	Used to find a locally attached Control Point Function (CPF).
CPDetect_C	See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for the encoding of this I/O.	Used to find a locally attached CPF.
CPDetect_D	See *Table 2-16: PMM Interface Pin Multiplexing* on page 57 for the encoding of this I/O.	Used to find a locally attached CPF.
Spare_Tst_Rcvr(9:0)	See *Table 2-31: Miscellaneous Pins* on page 75 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 NP2G 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 NP2G. The Clock_Core and the IEW Clock 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 NP2G

Once all configuration I/Os are set and the clocks are running, the NP2G can be reset using the $\overline{\text{Reset}}$ 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. At this point, the NP2G is ready to be booted. In addition to the $\overline{\text{Reset}}$ signal, the NP2G supports two other “soft” reset mechanisms: The Soft_Reset register allows the entire NP2G to be reset (just like a $\overline{\text{Reset}}$), and the PowerPC_Reset register allows the PowerPC core to be reset. A $\overline{\text{Reset}}$ or Soft_Reset causes the PowerPC_Reset register to activate and hold the PowerPC Core in reset until this register is cleared. Therefore, a $\overline{\text{Reset}}$ resets the entire NP2G 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 NP2G'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 NP2G 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. When using the last three methods, once the picocode instruction space is loaded, the BD_Override bit must be set to ‘1’ in the Boot Override Register (see *13.14.4 Boot Override Register (Boot_Override)* on page 463), which causes the GFH thread to start the code stored at address '0'.

11.4.2 Boot the PowerPC

There are two steps in the process of booting the NP2G'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 NP2G'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 NP2G'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 NP2G 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 11-3. 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 device). However, most of these structures can be tested by the diagnostics program to ensure proper operation. *Table 11-4* lists some of the structures that could be tested by this step.

Table 11-4. 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 Stores D0, D2, D3, D4	Read/Write	
Control Store D6	Read/Write	Coordinated with PowerPC code loading
Control Store H0-H1	Read/Write	
Counter Definition Table	Configure	Set up 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	
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 set up 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 11-5. Setup 2 Checklist

Register	Fields	Notes
Master Grant Mode	All	
Toggle Mode BCI 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	
FCB Free Queue Size Register (FCB_FQ_Max)	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. Note that external physical devices should be held in reset until the DMU configuration is completed.
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	
IEW	All	

11.8 Step 7: Hardware Initialization

Hardware Initialization allows the NP2G 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 11-6* need to be written.

Table 11-6. Hardware Initialization Checklist

Register	Fields	Notes
IEW Init	All	Only start the IEW initialization after the IEW has been configured.
Initialization	DMU set	Only start each DMU after its associated DMU Configuration has been set.
Initialization	Functional Island	Starts all other islands' self-initialization.

11.9 Step 8: Diagnostics 2

Diagnostics 2 determines if the NP2G is ready for operation and allows testing of data flow paths. The items listed in *Table 11-7* should be set up, checked, and/or tested.

Table 11-7. 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. IEW timeout = 20 ms E_EDS timeout = 15 ms
IEW Initialization	mode	Set the mode to operational.
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 the Egress side.
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 NP2G is ready to receive Guided Frames. This signal is activated by writing the NP2G 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^{2}C 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 NP2G for functional configuration. Items that can be configured include:

Table 11-8. Configure Checklist

Register	Fields	Notes
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 must be configured. Note that external physical devices should be held in reset until the DMU configuration is completed.
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.
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	
Target Port Data Storage Map	All	
Egress QCBs	All	
QD Accuracy	All	
SA Address Array	All	
Counter Definition Table	All	
Counters	All	Any Counters used by Counter Manager must be Read/Cleared.
Policy Manager Memory	All	Set up initial values for Policies.

11.12 Step 11: Initialization Complete

Once steps 1 through 10 are complete and all items on the checklists have been configured, the NP2G is ready for data traffic and the ports can be enabled.

12. Debug Facilities

12.1 Debugging Picoprocessors

The NP2G provides several mechanisms to facilitate debugging of the picoprocessors.

12.1.1 Single Step

Each thread of the NP2G 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 NP2G 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 NP2G supports RISCWatch through the JTAG interface.

RISCWatch is a hardware and software development tool for the embedded PowerPC. It provides processor control and source-level debugging features, including:

- On-chip debugging via IEEE 1149.1 (JTAG) interface
- Target monitor debugging
- Open-source real-time operating system-aware debugging
- Source-level and assembler debugging of C/C^{++} executable files
- Real-time trace support via the RISCTrace feature for the PowerPC 400 Series
- Network support that enables customers to remotely debug the systems they are developing
- 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 the customer is developing
- Ethernet to target JTAG interface hardware
- Multiple hosts supported
- Intuitive and easy-to-use windowed user interface

For more information, go to http://www-3.ibm.com/chips/techlib/techlib.nsf/products/RISCWatch_Debugger.

13. Configuration

The IBM PowerNP™ NP2G Network Processor must be configured after internal diagnostics have run. Configuration is performed by a control point function (CPF) that generates guided traffic to write configuration registers. These configuration registers are reset by the hardware to a minimal option set.

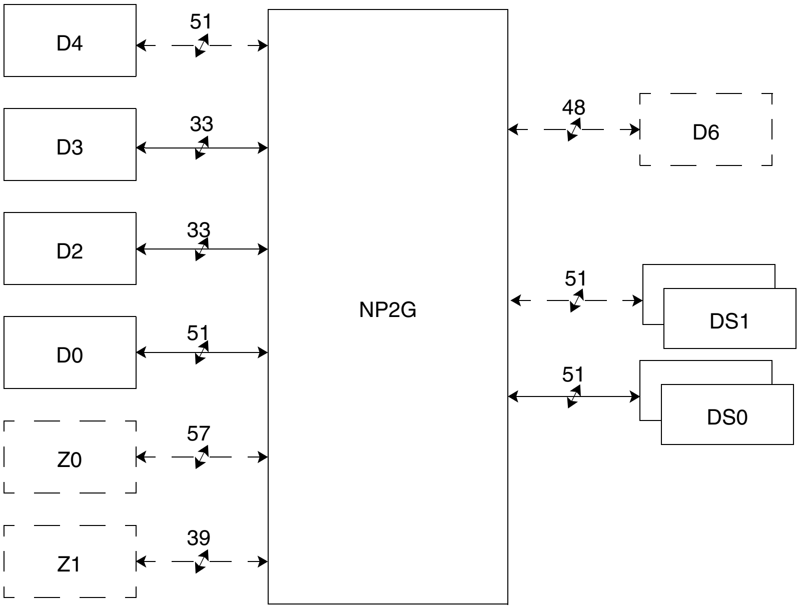
The following sections describe all configuration registers and their reset state. A base address and offset are provided.

13.1 Memory Configuration

The NP2G is supported by a number of memory subsystems as shown in *Figure 13-1*. These memory subsystems contain data buffers and controls used by the NP2G. The D0, D2, D3, D4, DS_0, and DS_1 subsystems are required to operate the NP2G base configuration. 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 NP2G 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 (Memory_Config)* (page 432). The bits in this register are set by hardware during reset to enable the base configuration.

Figure 13-1. NP2G Memory Subsystems

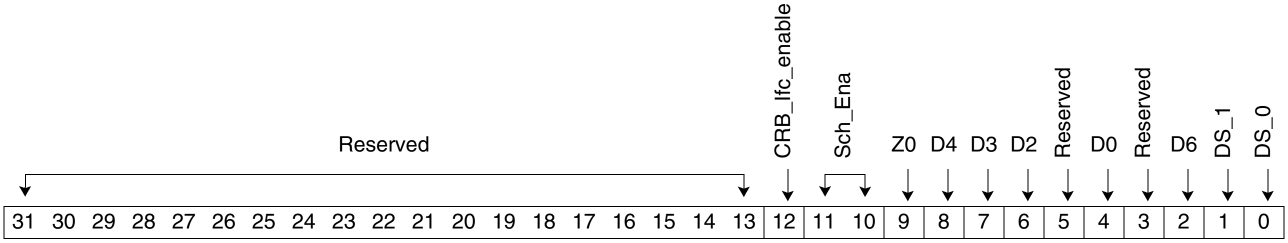


13.1.1 Memory Configuration Register (Memory_Config)

The *Memory Configuration Register (Memory_Config)* 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	Bits	Reset	Description
Reserved	31:13		Reserved.
CRB_Ifc_enable	12	0	When set to ‘1’, enables the Coprocessor Response Bus (CRB) coprocessor external interface. The Z0 Memory Subsystem Interface Enabled control (bit 9 of the *Memory Configuration Register (Memory_Config))* must also be set to ‘1’ to enable the CAB interface.
Sch_Ena	11:10	00	Scheduler Enabled control enables both egress scheduler and Z1 memory interface: 00 Scheduler disabled, Z1 interface disabled 01 Scheduler enabled, Z1 interface enabled 10 Scheduler disabled, Z1 interface enabled 11 Reserved When enabling the scheduler, 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) + free queue (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
Reserved	5	0	Reserved.
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

Field Name	Bits	Reset	Description
DS_1	1	1	DS_1 Memory Subsystem Interface Enabled control: 1 Interface enabled 0 Interface disabled
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 (DRAM_Parm)* controls the operation of the dynamic random-access memory (DRAM) used by the Embedded Processor Complex (EPC) and the Egress Enqueuer/Dequeuer/Scheduler (EDS). Each DRAM is controlled separately.

Access Type Read/Write

Base Address x‘A000 2400’

Base Address Offset 0

EPC_DRAM_Parms	Strobe_cntl	31 30
	Reserved	29
	CS_Refresh_Rate	28
	CS_Bank_R_W_Skip	27
	CAS_ Latency	26 25 24
	8_Bank_Ena	23
	11/$\overline{10}$	22
	Drive_Strength	21
	DLL_Disable	20
	DQS_Clamping_Ena	19
	DRAM_Size	18 17
	FET_Cntl_Ena	16
	Reserved	15
	D0_Width	14
EDS_DRAM_Parms	DS_D4_Refresh_Rate	13
	DS_Error_Checking_Disable	12
	D4_Error_Checking_Disable	11
	CAS_ Latency	10 9 8
	8_Bank_Ena	7
	11/$\overline{10}$	6
	Drive_Strength	5
	DLL_Disable	4
	DQS_Clamping_Ena	3
	DRAM_Size	2 1
	FET_Cntl_Ena	0

Base Address Offset 1

EPC_DRAM_Parms	Reserved	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14
D6_DRAM_Parms	D6_Refresh Rate	13
	D6_Bank_RW_Skp	12
	CAS_Latency	11 10 9
	11/10	8
	Drive_ Strength	7
	DLL_ Disable	6
	DQS_ Clamping_ Ena	5
	D6_DRAM_Size	4 3 2
	FET_Cntl_Ena	1
	D6_Parity_Mode	0

Base Address Offset 2

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

Base Address Offset 0

Field Name	Bits	Reset	Description
Strobe_cntl	31:30	00	Strobe control for double data rate (DDR) DRAM interfaces: 00 Each DDR DRAM interface uses one strobe (xx_DQS) for each byte of data lines. 01 D0, D2, D3 DDR DRAM interface use one strobe (xx_DQS) for each byte of data lines. DS0, DS1, and D4 DDR DRAM interface use one strobe (xx_DQS) for all data lines. 10 Reserved. 11 Reserved.
Reserved	29		Reserved.
CS_Refresh_Rate	28	1	CS_Refresh_Rate. Controls the refresh rate for D0, D2, and D3: 0 Double refresh rate (7.5 µs) 1 Normal refresh rate (15 µs)
CS_Bank_RW_Skp	27	0	Controls the number of banks that must be skipped within a DRAM access window when switching from a read to a write: 0 Skip 1 bank 1 Skip 2 banks The proper setting for this bit is related to the DRAM timing specifications and the CAS_Latency value. This control is for all Control Stores (D0, D2, D3, D6).
CS_Bank_RW_Skp	27	0	This bit controls the number of banks that must be skipped within a DRAM access window when switching from a read to a write: 0 Skip 1 bank. CAS_Latency (bits 26:24) must be set to ‘010’ to support this option. 1 Skip 2 banks. The proper setting for this bit is related to the times of the DRAM with respect to the CAS_Latency value. This control is for D0, D2, and D3 control stores.
CAS_Latency	26:24, 10:8	010	DRAM Column Address Strobe (CAS) Latency value corresponds to the DRAM’s read latency measured from the Column Address. (DDR DRAM Mode Register, bits 6:4; see the DRAM documentation for more information.) 000 - 001 Reserved 010 2 clock cycles (PC266A-compliant DRAMs) 011 3 clock cycles 100 - 101 Reserved 110 2.5 clock cycles (PC266B-compliant DRAMs) 111 Reserved
8_Bank_Enable	23, 7	0	Eight Bank Addressing Mode Enable control value. For proper operation, must be set to ‘0’.
11/$\overline{10}$	22, 6	1	Ten- or eleven-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. (DDR DRAM Extended Mode Register, bit 1, see the DRAM documentation for more information.) 0 Strong 1 Weak
DLL_Disable	20, 4	0	Delay-locked loop (DLL) Disable control. For proper operation, must be set to ‘0’. (DDR DRAM Extended Mode Register, bit 0, see the DRAM documentation for more information.)

Field Name	Bits	Reset	Description
DQS_Clamping_Ena	19, 3	0	Data Q Strobe (DQS) Clamping Enable control. For proper operation, must be set to ‘0’.
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 **Note:** D0 can also be configured to be double-wide (x32).
FET_Cntl_Ena	16, 0	0	Field-Effect Transistor (FET) Control Enable control. For proper operation, must be set to ‘0’.
Reserved	15		Reserved.
D0_Width	14	1	D0 Width control indicates if one or two DRAMs are used for the D0 Control Store (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.
DS_D4_Refresh_Rate	13	0	Refresh rate control for DS0, DS1 and D4: 0 Normal refresh rate (15 μs) 1 Double refresh rate (7.5μs)
DS_Error_Checking_Disable	12	0	Egress Data Store Error Checking Disable control. When this field is set to ‘1’, all DRAM error checking for the egress data store 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 Data Store (DS0, DS1) and D4 DRAMs: 00 4 x 1 M x 16 DDR DRAM x 2 (or 4 x 1 M x 32 DDR DRAM x 1) 01 4 x 2 M x 16 DDR DRAM x 2 (or 4 x 2 M x 32 DDR DRAM x 1) 10 4 x 4 M x 16 DDR DRAM x 2 (or 4 x 4 M x 32 DDR DRAM x 1) 11 4 x 512 K x 32 DDR DRAM x 1 The setting of this field affects the size of the extended stack queues (see *Section 4.5 Extended Stack Queues* on page 323) as follows: GQ 00 01 10 11 GTQ 48 K 96 K 192 K 24 K GPQ 48 K 96 K 192 K 24 K GFQ 96 K 192 K 384 K 48 K GR0 96 K 192 K 384 K 48 K GR1 96 K 192 K 384 K 48 K GB0 96 K 192 K 384 K 48 K GB1 96 K 192 K 384 K 48 K Discard 96 K 192 K 384 K 48 K

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
Reserved	31:14		Reserved.
D6_Refresh_Rate	13	0	This field controls the refresh rate for D6: 0 Normal refresh rate (15 µs) 1 Double refresh rate (7.5 µs)
D6_Bank_RW_Skp	12	0	This controls the number of banks that must be skipped within a DRAM access window when switching from a read to a write: 0 Skip 1 bank. CAS_Latency (bits11:9) must be set to '010' to support this option. 1 Skip 2 banks. The proper setting for this is related to the DRAM timing specifications and the CAS_Latency value. This control is for D6 only.
CAS_Latency	11:9	010	DRAM Column Address Strobe Latency value corresponds to the DRAM's read latency measured from the Column Address. (DDR DRAM Mode Register, bits 6:4; see the DRAM documentation for more information.) 000 - 001 Reserved 010 2 clock cycles (PC266A-compliant DRAMs) 011 3 clock cycles 100 - 101 Reserved 110 2.5 clock cycles (PC266B-compliant DRAMs) 111 Reserved
11/$\overline{10}$	8	0	Ten- or eleven-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	7	0	DRAM Drive Strength control. (DDR DRAM Extended Mode Register, bit 1, see the DRAM documentation for more information.) 0 Strong 1 Weak
DLL_Disable	6	0	DLL Disable control. For proper operation, must be set to '0'. (DDR DRAM Extended Mode Register, bit 0, see the DRAM documentation for more information.)
DQS_Clamping_Ena	5	0	DQS Clamping Enable control. For proper operation, must be set to '0'.
D6_DRAM_Size	4:2	110	Indicates the size of the D6 DRAM: 000 4x1Mx16 001 4x2Mx16 010 4x4Mx16 011 Reserved 100 4x4Mx4 101 4x8Mx4 110 4x16Mx4 111 Reserved
FET_Cntl_Ena	1	0	FET Control Enable control. For proper operation, must be set to '0':
D6_Parity_Mode	0	1	DRAM D6 Parity Mode Disable control value: 0 D6 interface supports an additional two DDR DRAMs which support byte parity. The hardware generates on write and checks parity on read. 1 D6 interface does not support parity.

Base Address Offset 2

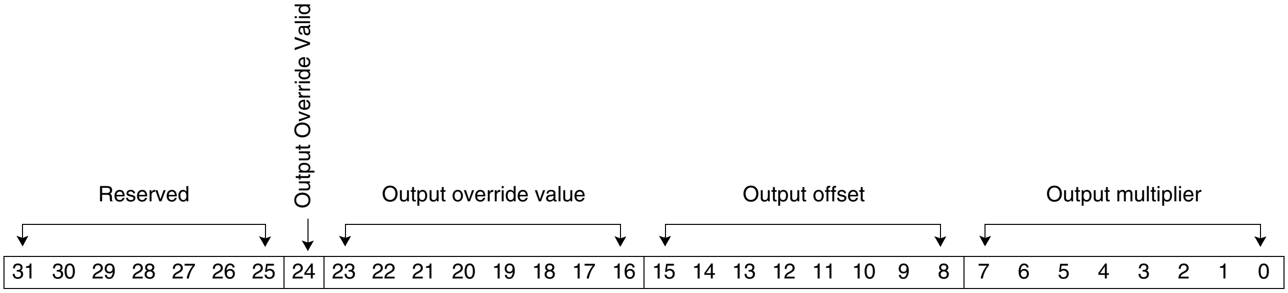
Field Name	Bit(s)	Reset	Description
Reserved	31:8		Reserved.
Auto_precharge	7:0	0	Configures the address bit used to flag the auto precharge operation for the attached DDR DRAM. When set to ‘0’, A10 is used. When set to ‘1’, A8 is used. Bit DRAM Interface 7 DS1 6 DS0 5 D6 4 D4 3 D3 2 D2 0 D0

13.1.3 Delay Calibration Registers

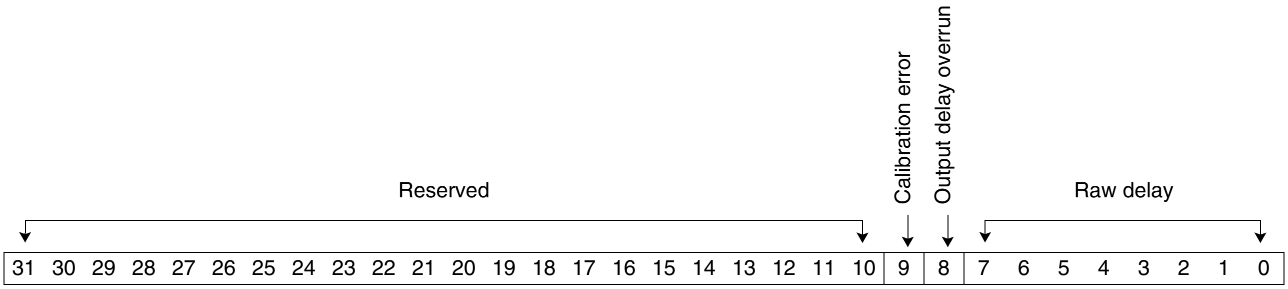
These registers are used to modify the interface timings for the DDR interfaces. These registers can be used to initialize the output multiplier fields in the following offsets to x’6A’. Otherwise, these registers should be modified only when directed by an IBM field engineer.

Access Type	Base Address Offset 0 Base Address Offset 2 Base Address Offset 4 Base Address Offset 6 Base Address Offset 8 Base Address Offset 10 Base Address Offset 12 Base Address Offset 15	Read/Write
	Base Address Offset 1 Base Address Offset 3 Base Address Offset 5 Base Address Offset 7 Base Address Offset 9 Base Address Offset 11 Base Address Offset 13	Read Only
Base Address	D0_DDR_DCR	x‘A080 0010’
	D2_DDR_DCR	x‘A080 0040’
	D3_DDR_DCR	x‘A080 0080’
	D4_DDR_DCR	x‘A080 0100’
	D6_DDR_DCR	x‘A080 0200’
	DS0_DDR_DCR	x‘A080 0400’
	DS1_DDR_DCR	x‘A080 0800’

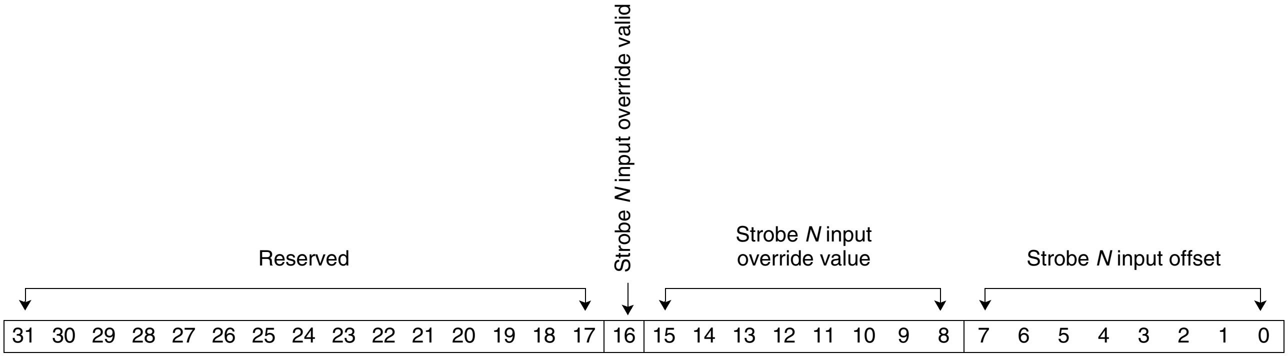
Base Address Offset 0 (R/W) (D0, D2, D3, D4, D6, DS0, and DS1)



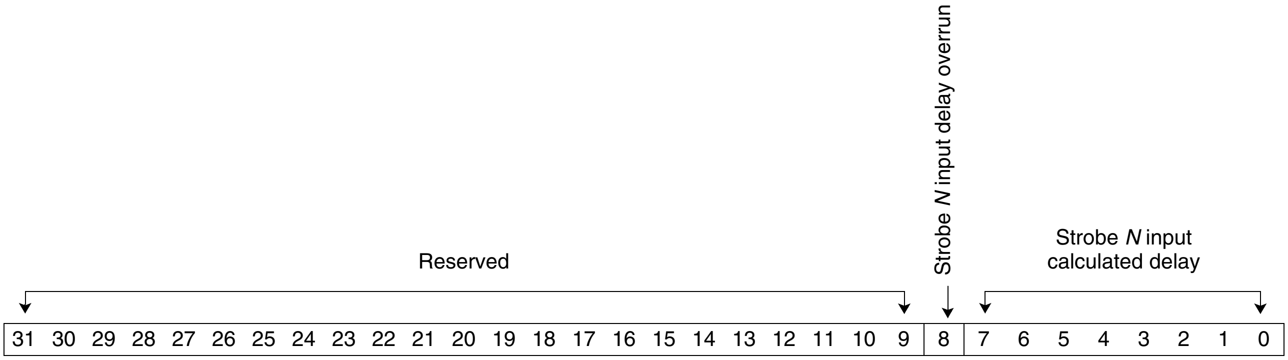
Base Address Offset 1 (RO) (D0, D2, D3, D4, D6, DS0, and DS1)



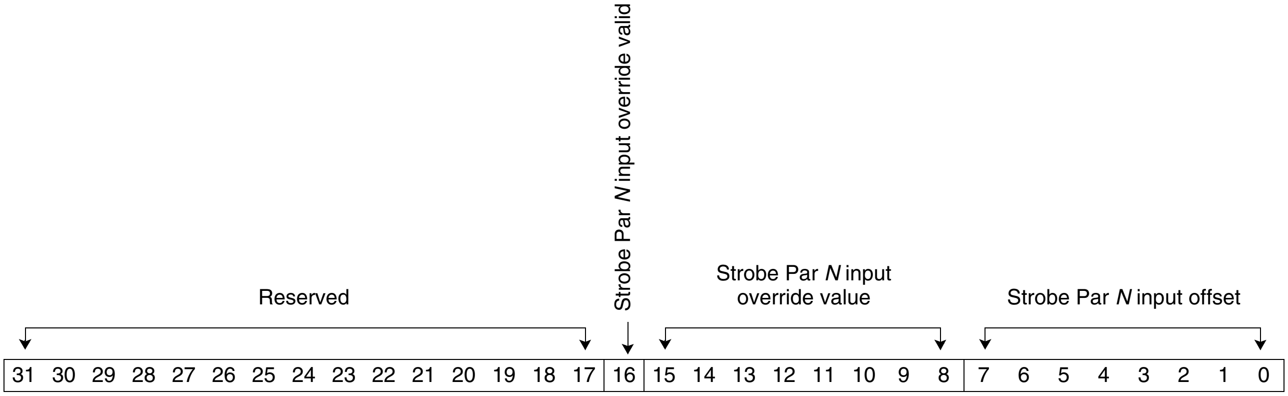
Base Address Offset 2, *N* = 0 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 4, *N* = 1 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 6, *N* = 2 (R/W) (D0, D4, D6, DS0, DS1)
Base Address Offset 8, *N* = 3 (R/W) (D0, D4, D6, DS0, DS1)



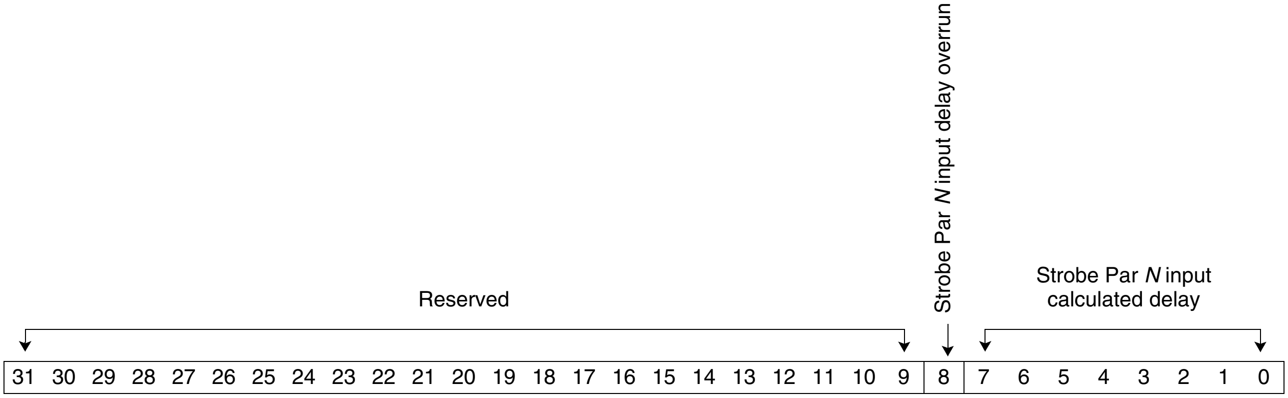
Base Address Offset 3, *N* = 0 (RO) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 5, *N* = 1 (RO) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 7, *N* = 2 (RO) (D0, D4, D6, DS0, DS1)
Base Address Offset 9, *N* = 3 (RO) (D0, D4, D6, DS0, DS1)



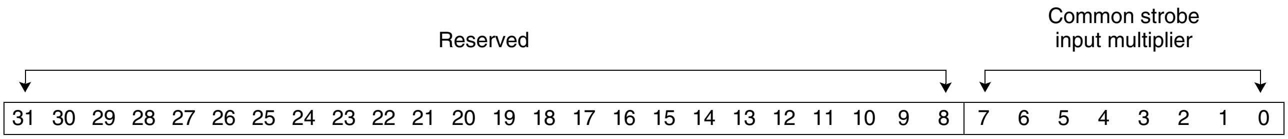
Base Address Offset 10, *N* = 0 (R/W) (D6)
Base Address Offset 12, *N* = 1 (R/W) (D6)


Reserved
Strobe Par *N* input override valid
Strobe Par *N* input override value
Strobe Par *N* input offset
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17
16
15 14 13 12 11 10 9 8
7 6 5 4 3 2 1 0

Base Address Offset 11, *N* = 0 (RO) (D6)
Base Address Offset 13, *N* = 1 (RO) (D6)



Base Address Offset 15 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)



Base Address Offset 0 (R/W) (D0, D2, D3, D4, D6, DS0, and DS1)

Field Name	Bit(s)	Reset	Description
Reserved	31:25		Reserved.
Output Override Valid	24	0	Output override valid.
Output override value	23:16	0	Output override value.
Output offset	15:8	0	Output offset.
Output multiplier	7:0	x‘35’	Output multiplier. Prior to initializing the DDR DRAMs, this field must be written to a value of x‘6A’.

Base Address Offset 1 (RO) (D0, D2, D3, D4, D6, DS0, and DS1)

Field Name	Bit(s)	Reset	Description
Reserved	31:10		Reserved.
Calibration error	9	0	Calibration error.
Output delay overrun	8	0	Output delay overrun.
Raw delay	7:0	0	Raw delay.

Base Address Offset 2, *N* = 0 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 4, *N* = 1 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 6, *N* = 2 (R/W) (D0, D4, D6, DS0, DS1)
Base Address Offset 8, *N* = 3 (R/W) (D0, D4, D6, DS0, DS1)

Field Name	Bit(s)	Reset	Description
Reserved	31:17		Reserved.
Strobe *N* input override valid	16	0	Strobe *N* input override valid.
Strobe *N* input override	15:8	0	Strobe *N* input override value.
Strobe *N* input offset	7:0	0	Strobe *N* input offset.

Base Address Offset 3, *N* = 0 (RO) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 5, *N* = 1 (RO) (D0, D2, D3, D4, D6, DS0, DS1)
Base Address Offset 7, *N* = 2 (RO) (D0, D4, D6, DS0, DS1)
Base Address Offset 9, *N* = 3 (RO) (D0, D4, D6, DS0, DS1)

Field Name	Bit(s)	Reset	Description
Reserved	31:9		Reserved.
Strobe *N* input delay overrun	8	0	Strobe *N* input delay overrun.
Strobe *N* input calculated delay	7:0	0	Strobe *N* input calculated delay.

Base Address Offset 10, *N* = 0 (R/W) (D6)
Base Address Offset 12, *N* = 1 (R/W) (D6)

Field Name	Bit(s)	Reset	Description
Reserved	31:17		Reserved.
Strobe Par *N* input override valid	16	0	Strobe Par *N* input override valid.
Strobe Par *N* input override value	15:8	0	Strobe Par *N* input override value.
Strobe Par *N* input offset	7:0	0	Strobe Par *N* input offset.

Base Address Offset 11, *N* = 0 (RO) (D6)
Base Address Offset 13, *N* = 1 (RO) (D6)

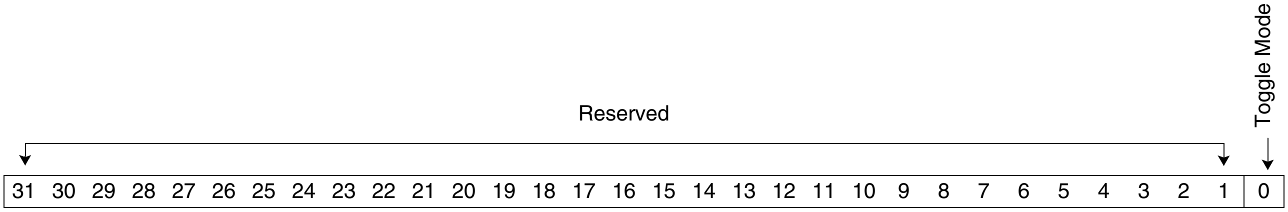
Field Name	Bit(s)	Reset	Description
Reserved	31:9		Reserved.
Strobe Par *N* input delay overrun	8	0	Strobe Par *N* input delay overrun.
Strobe Par *N* input calculated delay	7:0	0	Strobe Par *N* input calculated delay.

Base Address Offset 15 (R/W) (D0, D2, D3, D4, D6, DS0, DS1)

Field Name	Bit(s)	Reset	Description
Reserved	31:8		Reserved.
Common strobe input multiplier	7:0	x‘6A’	Common strobe input multiplier.

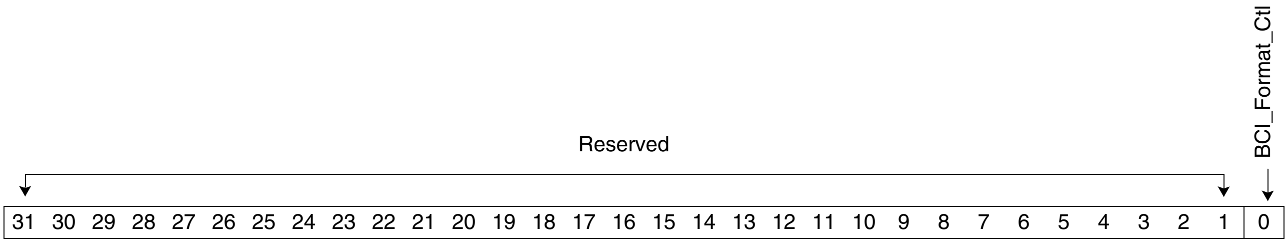
13.2 Toggle Mode Register

Access Type	Read/Write
Base Address	x'A000 0411'



13.3 BCI Format Register

Access Type	Read/Write
Base Address	x'A000 0412'



Base Address Offset 1

Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved.
Toggle Mode	0	0	Enables the toggle mode. 0 Enables egress data store standard mode operation. Places frame data into DS0, DS1, or both data stores. Data storage unit (DSU) information fields indicate the following: Two-Bit DSU Information Field 00 Reserved 01 DS0 10 DS1 11 DS0 and DS1 One-Bit DSU Information Field 0 DS0 1 DS1 1 Enables egress data store toggle mode operation. Places frame data into DS0 and DS1 according to the twin format. DSU information fields indicate the following: Two-Bit DSU Information Field 00 Invalid 01 Twin format 0 10 Twin format 1 11 Invalid One-Bit DSU Information Field 0 Twin format 0 1 Twin format 1

Base Address Offset 2

Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved.
BCI_Format_Ctl	0	0	Controls the format of the byte count information (BCI) used in the egress frame control block page (FCBPage). The BCI is located at word 1 (FCBPage byte offsets 4 through 7) of the FCBPage. See *Section 7.4.4 Enqueue Coprocessor* on page 223. 0 Compact format: Bits Description 31:20 Reserved 19:14 Starting byte 13:6 Number of data buffers (weight) 5:0 Ending byte 1 Expanded format: Bits Description 31:22 Reserved 21:16 Starting byte 15:8 Number of data buffers (weight) 7:6 Reserved 5:0 Ending byte

13.3.1 Toggle Mode

The toggle mode provides increased bandwidth to egress data stores by changing the data storage method. The toggle mode is activated by configuring the corresponding field in the *Toggle Mode Register* (page 444). All Data Mover Unit (DMU) modes are supported.

When the toggle mode is inactive, frame data is assigned to DS0, DS1, or both data stores. Bandwidth limitations occur when there is a consistent requirement for data to be stored in both data stores, such as with multicast and guided traffic.

The toggle mode changes the data storage method by defining a “twin” format for use by all twins assigned to a frame. Unicast, multicast, and guided traffic frames are all assigned in the same manner. When the toggle mode is active, multicast and guided frames are assigned a twin format so that they are not written into both data stores. DSU settings found in the frame header are ignored; the twin format controls which data store is used for reading and writing.

The twin format is assigned when the first frame buffer is loaded into the data store. For unpacked frames, the assignment toggles between formats at each start of frame. For packed frames, the assignment is based on the current write window.

Two twin formats are defined:

- *Twin format 0.* The even buffer is located in DS0 and the odd buffer is located in DS1.
- *Twin format 1.* The even buffer is located in DS1 and the odd buffer is located in DS0.

For packed frames, assignment is based on where the current frame is written. This definition allows the entire packed cell (end of current frame and start of next frame) to be stored during the current write window:

Current Frame Data Store	Next Frame Format
0	0
1	1

13.3.1.1 DSU information

The NP2G passes the DSU information that originates from the frame header to the general queues (GR0, GR1, GB0, GB1, GFQ, GPQ, and GTQ), dispatch unit, data store coprocessor, FCBPage, discard queue, and common FCB. With the exception of the frame header DSU field, the various fields employed for this purpose are reused (when the toggle mode is active) to indicate the frame's assigned format.

When the toggle mode is active, a two-bit DSU information field is defined as follows:

00	Invalid
01	Twin format 0
10	Twin format 1
11	Invalid

When the toggle mode is active, a one-bit DSU information field is defined as follows:

0	Twin format 0
1	Twin format 1

An example of a one-bit DSU field is the Type(0) field found in the general queues (GR0, GR1, GB0, GB1, GFQ, GPQ, and GTQ). To maintain the frame order in general data queues, GR1 or GB1 is always used.

Note: Dispatch port configuration table (PCT) entries 48 and 49 should be identical when the toggle mode is enabled because no information can be inferred about the frame other than the twin format.

Because the frame header DSU field is not used when the toggle mode is active, the egress FCBPage DSU field and the Disp_DSU field (both two-bit DSU information fields) are initialized based on the value of Type(0):

Type(0)	FCBPage DSU, Disp_DSU
0	01
1	10

The DSControl field (a two-bit DSU information field) in the WREDS, RDEDS, RDMOREE, and EDIRTY data store coprocessor commands indicate data store use. When the toggle mode is active, the DSControl field determines the format of the twins being written:

00	Uses the Disp_DSU value captured at dispatch
01	Forces twin format 0
10	Forces twin format 1
11	Invalid

Disp_DSU conforms to the definition of a two-bit DSU information field.

For the Egress General Data Queue (E-GDQ), Type(1:0) indicates the twin format used and conforms to the two-bit DSU field definition above.

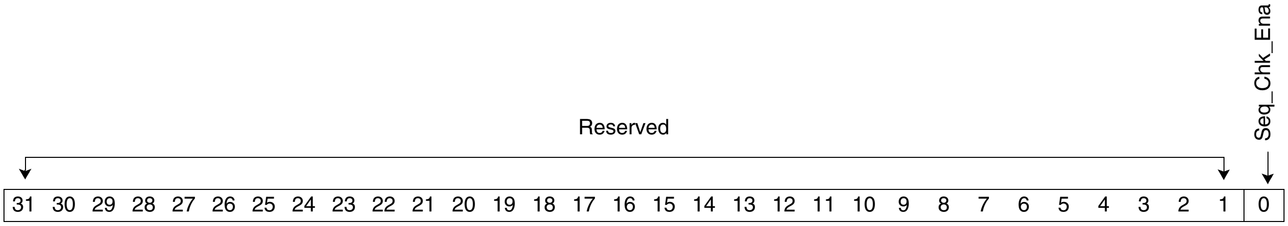
13.3.1.2 ENQE Command and Qclass 5 or 6

When the toggle mode is inactive, the general data queue selected (GR0, GR1, GB0, or GB1) is based on the value of QID(2:0), where QID(0) is set to the DSUSel value. When the toggle mode is active, GR1 or GB1 is used. This matches the selection made when the enqueue to the general data queues is performed as described in the previous section.

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 ensures 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'

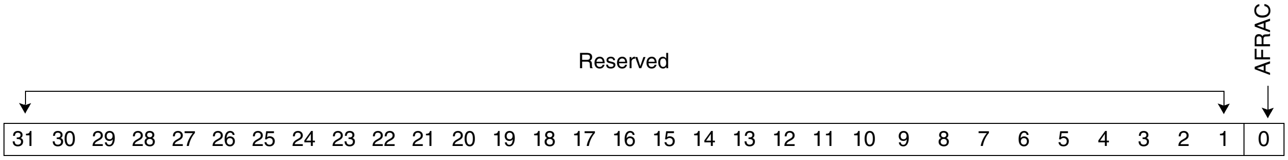


Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved.
Seq_Chk_Ena	0	1	Sequence Check Enable control: 0 Sequence checking disabled 1 Sequence checking enabled for the E_EDS

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.

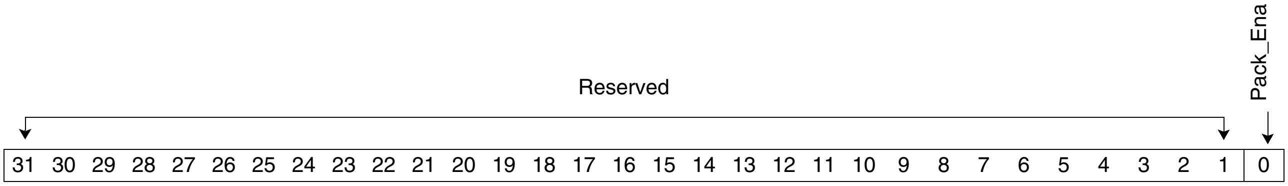
13.6 Packing Registers

13.6.1 Packing Control Register (Pack_Ctrl)

This configuration register is used to enable the transmission of packed cells through the IEW. When enabled, the ingress Cell Data Mover (I-CDM) determines if there is available space in the current cell being prepared for transmission and if there is an available packing candidate. A frame can be packed only if it is unicast and is longer than 48 bytes. The packed frame starts on the next quadword (QW) boundary following the last byte of the current frame. This control does not affect the ability of the E_EDS to receive packed cells.

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

13.6.2 Packing Delay Register (Pack_Dly)

The *Packing Delay Register (Pack_Dly)* is used to modify the ingress scheduler actions by the use of programmable wait periods based on source port (SP) media speed, as determined by the DM_Bus_mode setting (see *Data Mover Unit (DMU) Configuration Registers* [page 485]). During processing of a frame for transmission to the egress side, when it is determined that the next service will be the last for the frame and if there is no packing candidate at that time, then the selected run queue is not included in subsequent service selections. A timer is used per run queue to keep the queue out of the service selection until either the timer expires or a packing candidate (the corresponding start-of-frame [SOF] ring goes from empty to nonempty) is available.

Frames from the wrap DMU are not affected by this mechanism. Multicast frames (which include guided traffic) and the discard queues are not affected by this mechanism.

This function is disabled by setting the packing delay to ‘0’.

Access Type Read/Write

Base Address x'A020 8000'

Reserved												OC3_Delay				OC12_Delay				FastEn_Delay				Gb_Delay				Reserved			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Field Name	Bit(s)	Reset	Description
Reserved	31:20		Reserved.
OC3_Delay	19:16	0	Maximum packing delay in units of 3840 ns.
OC12_Delay	15:12	0	Maximum packing delay in units of 960 ns.
FastEn_Delay	11:8	0	Maximum packing delay in units of 4800 ns.
Gb_Delay	7:4	0	Maximum packing delay in units of 480 ns.

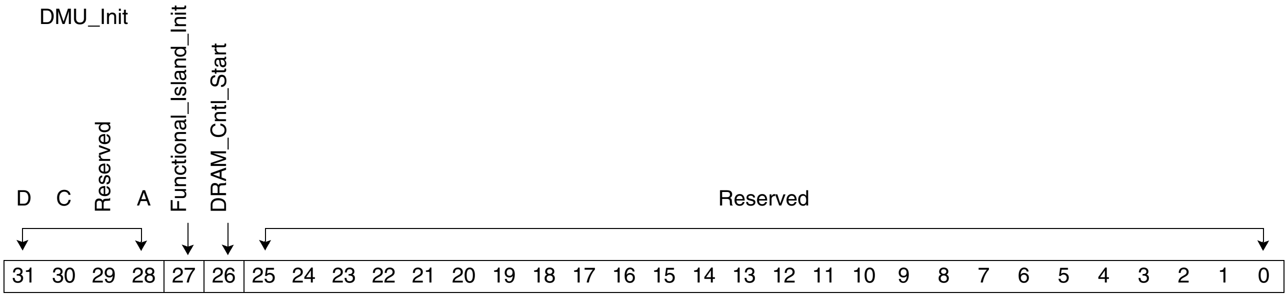
13.7 Initialization Control Registers

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 *Initialization Done Register (Init_Done)* (page 452). Once a functional island has been initialized, changing the state of these bits will no longer have any effect until a reset occurs.

Access Type Read/Write

Base Address x‘A000 8100’



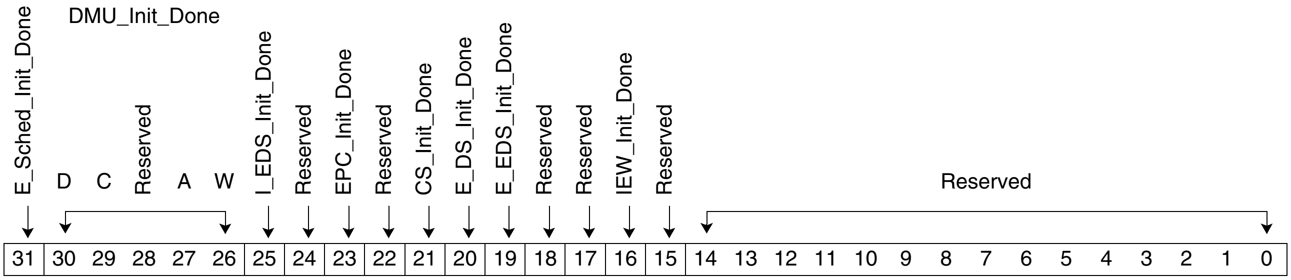
Field Name	Bit(s)	Reset	Description
DMU_Init	31, 30, 28	000	Data Mover Unit Initialization control individually initializes each DMU.
Functional_Island_Init	27	0	Functional Island Initialization control: 0 No operation (nop) 1 Functional islands start hardware initialization. Completion of hardware initialization is reported in the *Initialization Done Register (Init_Done).*
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 CS_Init_Done and E-DS Init Done bits of the *Initialization Done Register (Init_Done).*
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 Guided Frame Handler (GFH) reads this register during the initialization of the NP2G.

Access Type Read Only

Base Address x‘A000 8200’

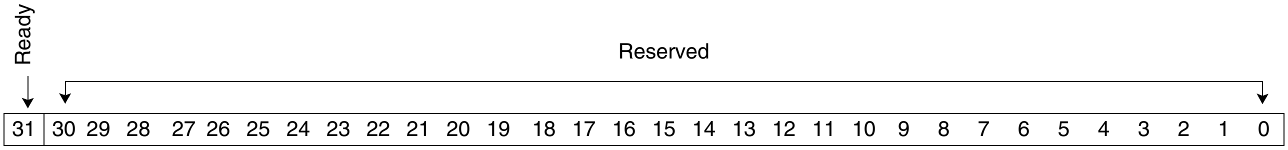


Field Name	Bit(s)	Reset	Description
E_Sched_Init_Done	31	0	Set to ‘1’ by the hardware when the egress scheduler hardware completes its initialization.
DMU_Init_Done	30, 29, 27, 26	0	Set to ‘1’ by the hardware when the DMU hardware has completed its initialization. Bit DMU 30 D 29 C 27 A 26 Wrap
I_EDS_Init_Done	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.
CS_Init_Done	21	0	Set to ‘1’ by the hardware when the Control Store's DRAM controller 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.
IEW_Init_Done	16	0	Set to ‘1’ by the hardware when the IEW completes its initialization.
Reserved	14:0		Reserved.

13.8 NP2G 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 NP2G has been initialized and is ready to receive Guided Traffic: 0 NP2G not ready 1 NP2G ready
Reserved	30:0		Reserved.

13.9 Phase-Locked Loop Registers

13.9.1 Phase-Locked Loop Fail Register (PLL_Lock_Fail)

Access Type Read Only

Base Address x‘A000 0220’

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

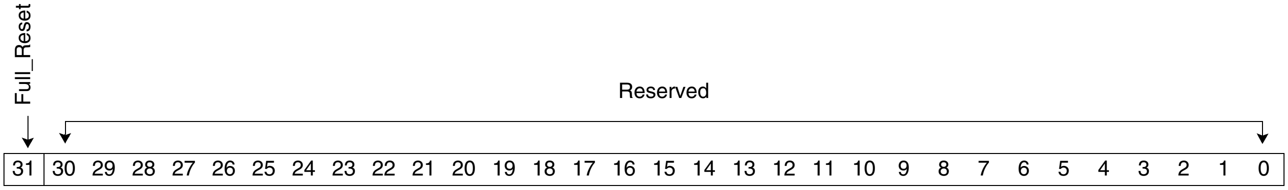
Field Name	Bit(s)	Reset	Description
Reserved	31:7		Reserved.
Pllc_lock_failed	6		Pllc_lock_failed. 0 Pllc has stayed correctly locked continuously since the last reset. 1 Pllc has lost lock. (Lock was lost any time since the last reset.)
Pllb_lock_failed	5		Pllb_lock_failed. 0 Pllb has stayed correctly locked continuously since the last reset. 1 Pllb has lost lock. (Lock was lost any time since the last reset.)
Plla_lock_failed	4		Plla_lock_failed. 0 Plla has stayed correctly locked continuously since the last reset. 1 Plla has lost lock. (Lock was lost any time since the last reset.)
PLL_C_Lock	3		Current status of lock indicator of the Core Clock PLL: 0 Phase/frequency lock 1 Phase/frequency seek
PLL_B_Lock	2		Current status of lock indicator of the IEW-B PLL: 0 Phase/frequency lock 1 Phase/frequency seek
PLL_A_Lock	1		Current status of lock indicator of the IEW-A PLL: 0 Phase/frequency lock 1 Phase/frequency seek
Fail	0		Phase-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

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 NP2G hardware via the picocode. This is the same full reset function provided by the Reset I/O. 0 Reserved 1 NP2G performs an internal hardware reset
Reserved	30:0		Reserved.

13.11 Ingress Free Queue Threshold Configuration

13.11.1 BCB_FQ Threshold Registers

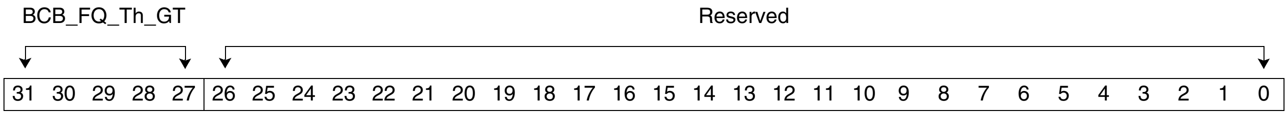
The value of the queue count in the BCB_FQ Control Block is continuously compared to the values each of the three threshold registers contains. The result of this comparison affects the NP2G’s flow control mechanisms. The values in these registers must be chosen such that BCB_FQ_Th_GT < BCB_FQ_Th_0 < 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'

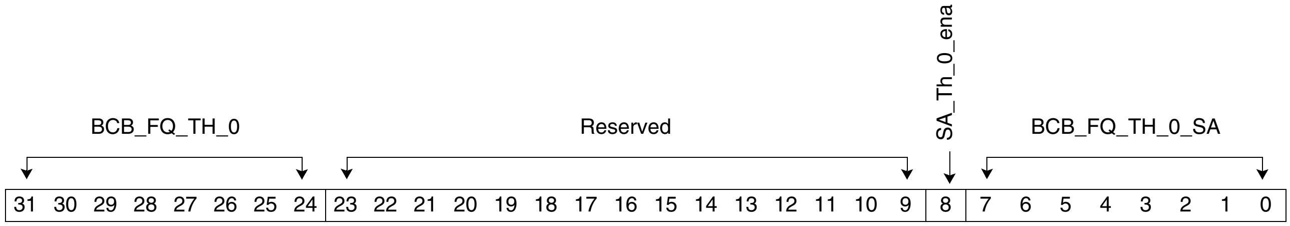


Field Name	Bit(s)	Reset	Description
BCB_FQ_Th_GT	31:27	x‘08’	Buffer Control Block (BCB) Free Queue Threshold Guided Traffic (GT) value is measured in units of individual buffers. For example, a threshold value of x‘01’ represents a threshold of one buffer.
Reserved	26:0		Reserved.

13.11.3 BCB_FQ_Threshold_0 Register (BCB_FQ_TH_0)

Access Type Read/Write

Base Address x‘A000 1010’

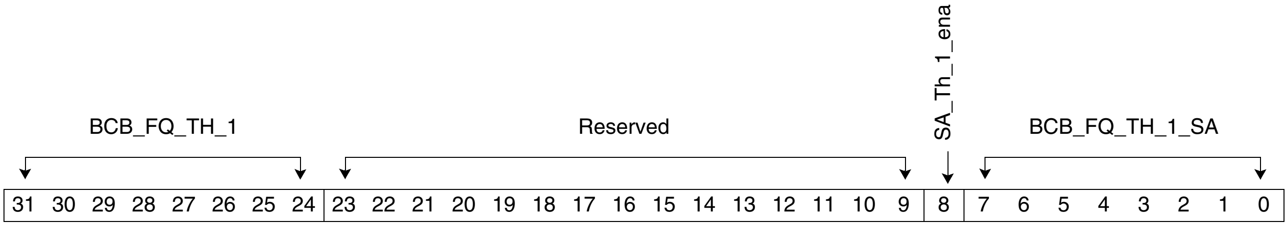


Field Name	Bit(s)	Reset	Description
BCB_FQ_TH_0	31:24	x‘01’	BCB Free Queue Threshold 0 value, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. The Ingress EDS reads this field to determine when to perform a discard action. Violation of this threshold (BCB queue count is less than the value indicated by 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. This discard action continues until the BCB queue count is greater than the value indicated by the BCB_FQ_TH_0_SA value when SA_Th_0_ena is set to ‘1’, or when the BCB free queue count is greater than the value indicated by the BCB_FQ_TH_0 value when SA_Th_0_ena is set to ‘0’.
Reserved	23:9		Reserved.
SA_Th_0_ena	8	0	Stop action Threshold 0 enable. 0 The discard action started due to the BCB free queue count dropping below the value indicated by BCB_FQ_TH_0 continues until the BCB free queue count is greater than the value indicated by BCB_FQ_TH_0. 1 The discard action started due the BCB free queue count dropping below the value indicated by BCB_FQ_TH_0 continues until the BCB free queue count is greater than the value indicated by BCB_FQ_TH_0_SA.
BCB_FQ_TH_0_SA	7:0	x‘01’	BCB Free queue stop action threshold, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. When SA_Th_0_ena is set to ‘1’, this value is used to stop the discard action started due to the BCB free queue count dropping below the value indicated by BCB_FQ_TH_0.

13.11.4 BCB_FQ_Threshold_1 Register (BCB_FQ_TH_1)

Access Type Read/Write

Base Address x‘A000 1020’

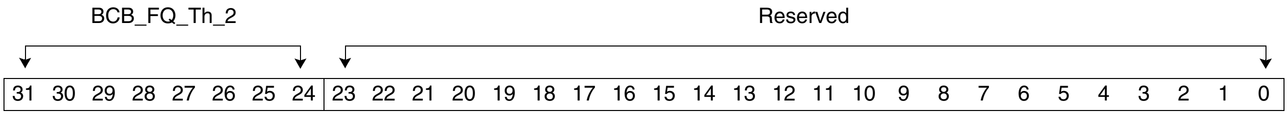


Field Name	Bit(s)	Reset	Description
BCB_FQ_TH_1	31:24	x‘01’	BCB Free Queue Threshold 1 value, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. The Ingress EDS reads this field to determine when to perform a discard action. Violation of this threshold (BCB queue count is less than the value indicated by this threshold) sends an interrupt to the EPC. 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 allocate buffers as needed. This discard action continues until the BCB queue count is greater than the value indicated by the BCB_FQ_TH_1_SA value when SA_Th_0_ena is set to ‘1’, or when the BCB free queue count is greater than the value indicated by the BCB_FQ_TH_1 value when SA_Th_1_ena is set to ‘0’.
Reserved	23:9		Reserved.
SA_Th_1_ena	8	0	Stop action Threshold 1 enable. 0 The discard action started due to the BCB free queue count dropping below the value indicated by BCB_FQ_TH_1 continues until the BCB free queue count is greater than the value indicated by BCB_FQ_TH_1. 1 The discard action started due the BCB free queue count dropping below the value indicated by BCB_FQ_TH_1 continues until the BCB free queue count is greater than the value indicated by BCB_FQ_TH_1_SA.
BCB_FQ_TH_1_SA	7:0	x‘01’	BCB Free queue stop action threshold, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. When SA_Th_1_ena is set to ‘1’, this value is used to stop the discard action started due to the BCB free queue count dropping below the value indicated by BCB_FQ_TH_1.

13.11.5 BCB_FQ_Threshold_2 Register (BCB_FQ_Th_2)

Access Type Read/Write

Base Address x‘A000 1040’



Field Name	Bit(s)	Reset	Description
BCB_FQ_Th_2	31:24	x'01'	BCB Free Queue Threshold 2 value, as measured in units of 16 buffers. For example, a threshold value of x‘01’ represents a threshold of 16 buffers. Violation of this threshold (BCB queue count is less than this threshold) sends an interrupt to the EPC.
Reserved	23:0		Reserved.

13.12 I-GDQ Threshold Register (I-GDQ_Th)

During ingress data movement for a frame being received, an Ingress General Data Queue (I-GDQ) threshold is tested and, if exceeded, the frame may be discarded. The discard action is to perform packet discards. New data buffers are not allocated for new frame traffic, but frames already started continue to allocate buffers as needed and will be enqueued as necessary. This is similar to actions taken with BCB_FQ_TH_1.

Discards due to this threshold being violated are included in the count maintained in the NFD_Cnt field of the *Ingress Port Control Blocks Register (PCB and Wrap PCB)* (page 99). For debug and tuning, a global count of discards due to this threshold is maintained in the *I-GDQ Threshold Discard Count (I-GDQ_Th_Cnt) Register* (page 98).

Access Type Read/Write

Base Address x‘A000 1100’

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

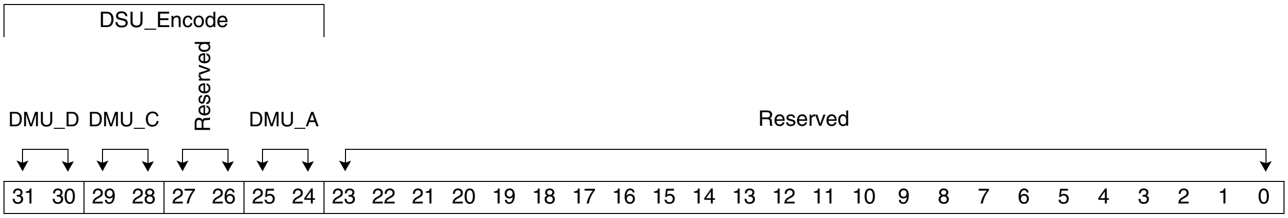
Field Name	Bit(s)	Reset	Description
Reserved	31:28		Reserved.
gdq_discard_th_hi	27:16	x'800'	Ingress GDQ high threshold. When the I-GDQ queue exceeds this threshold, the frame currently being serviced for data movement into the ingress data store may be discarded. The discard action is to perform new packet discards. New data buffers are not allocated for new frame traffic, but frames already started continue to allocate buffers as needed and will be enqueued as necessary.
Reserved	15:12		Reserved.
gdq_discard_th_lo	11:0	x'800'	If discards are occurring because the Ingress GDQ high threshold was exceeded, the number of entries in the GDQ queue must drop below this threshold before enqueues to the GDQ queue can resume.

13.13 Ingress Target DMU Data Storage Map Register (I_TDMU_DSU)

This register defines the egress Data Storage Units (DSUs) 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 Reserved 24:25 DMU_A
Reserved	23:0		Reserved.

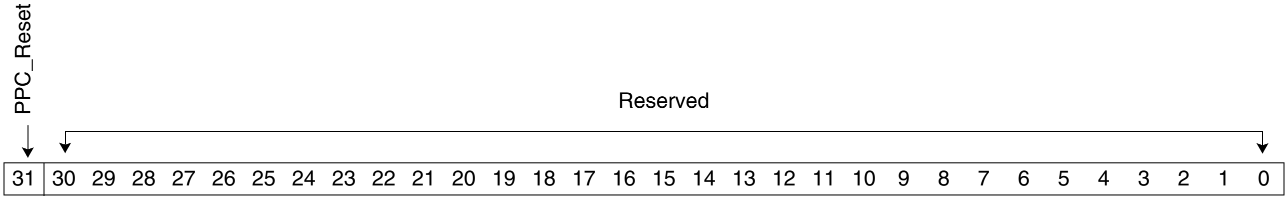
13.14 Embedded Processor Complex Configuration

13.14.1 PowerPC Core Reset Register (PowerPC_Reset)

This register contains a control value used to hold the PowerPC (PPC) 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 PowerPC core in a reset state when set to ‘1’. The rest of the PowerPC 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.

13.14.2 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 Registers (Boot_Redir_Inst)* are accessed from the CAB interface. These registers provide capacity for eight instructions for Processor Local Bus (PLB) addresses x‘FFFF FFE0’ - x‘FFFF FFFC’ These instructions redirect the PowerPC to boot from a location in the PowerPC’s DRAM and are configured before the PPC_Reset is cleared.

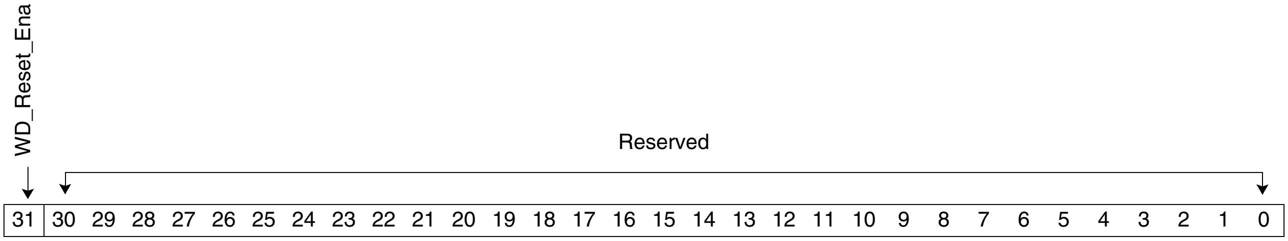
Access Type	Read/Write
Base Address	x‘3800 0110’ - x‘3800 0117’

13.14.3 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’



13.14.4 Boot Override Register (Boot_Override)

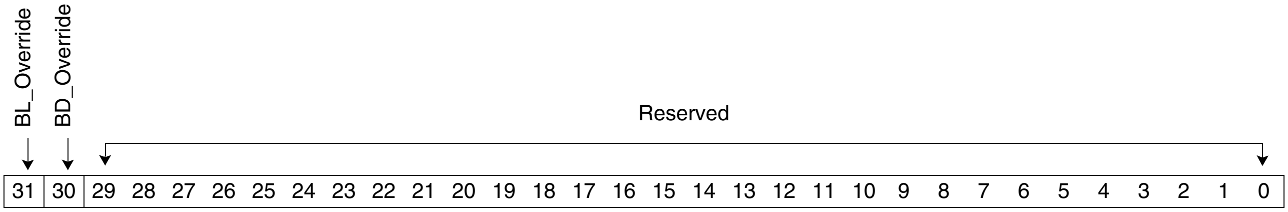
With the proper setting of the Boot_Picocode I/O (see *Table 2-31: Miscellaneous Pins* on page 75*),* this register provides boot sequence control.

When the Boot_Picocode I/O is held to ‘0’, boot control is given over to the Serial/Parallel Manager (SPM) interface hardware. When the SPM completes its boot actions, a power-on reset (POR) interrupt is set (Interrupt vector 1, bit 0) which causes the GFH to start execution.

When the Boot_Picocode I/O is held to ‘1’, boot control is given over to an external source, either a host on the peripheral component interconnect (PCI) or the ePPC. When boot is completed, a CAB write is required to set BD_Override to ‘1’, which causes a POR interrupt (Interrupt vector 1, bit 0), causing the GFH to start execution.

Access Type Read/Write

Base Address x‘A000 8800’

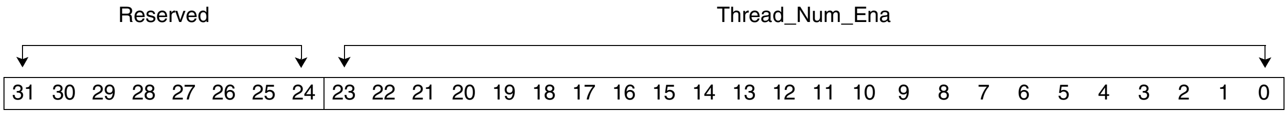


Field Name	Bit(s)	Reset	Description
BL_Override	31	See description	Set to the value of the boot_picocode I/O during reset: 0 Boot code is loaded by the SPM interface state machine. 1 Boot code is loaded by intervention of software. The boot code can be loaded using the CABWatch interface, using the PCI 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.

13.14.5 Thread Enable Register (Thread_Enable)

This register contains control information used to enable or disable each thread.

Access Type	Read/Write	Bits 23:1
	Read Only	Bit 0
Base Address	x‘A000 8020’	

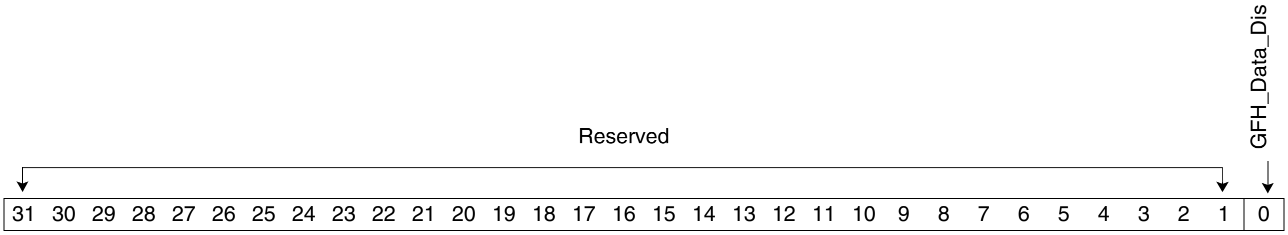


Field Name	Bit(s)	Reset	Description
Reserved	31:24		Reserved.
Thread_Num_Ena (23:1)	23:1	0	Thread enable: 0 Disabled 1 Corresponding thread enabled for use
Thread_Num_Ena 0	0	1	Thread 0 (the GFH) is always enabled and cannot be disabled through this bit.

13.14.6 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.

13.14.7 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 dispatch control block (DCB).

Access Type Read/Write

Base Address x'2440 0C40'

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

Field Name	Bit(s)	Reset	Description
I_Max_DCB	31:27	x'10'	Maximum number of ingress frames allowed service in the dispatch control block.
Reserved	26:0		Reserved.

13.14.8 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 dispatch control block.

Access Type Read/Write

Base Address x'2440 0C50'

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

Field Name	Bit(s)	Reset	Description
E_Max_DCB	31:27	x'10'	Maximum number of ingress frames allowed service in the dispatch control block.
Reserved	26:0		Reserved.

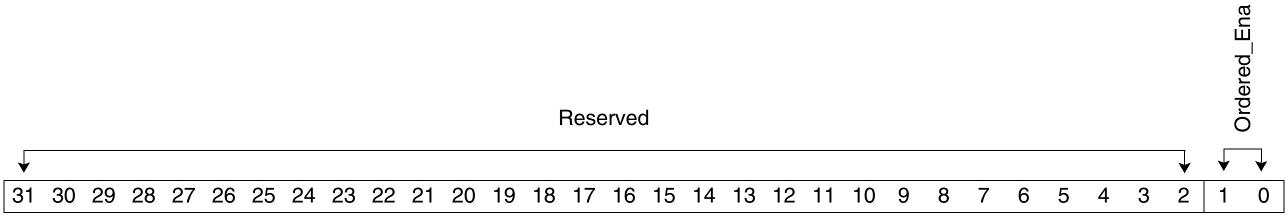
13.14.9 Ordered Semaphore Enable Register (Ordered_Sem_Ena)

This register enables ordered semaphore operation in the NP2G. A thread may have two semaphores locked at a time, with the restriction that only one of them may be ordered. When ordered semaphores are disabled, a thread may hold locks on two unordered semaphores at a time.

Note: Failure to follow these restrictions will result in unpredictable operation.

Access Type Read/Write

Base Addresses x'2500 0180'



Field Name	Bit(s)	Reset	Description
Reserved	31:2		Reserved.
Ordered_Ena	1:0	0	00 Ordered semaphores disabled. 01 Ordered semaphores enabled for Ordered Semaphore ID Queue 0. 10 Ordered semaphores enabled for Ordered Semaphore ID Queue 1. 11 Ordered semaphores enabled for both Ordered Semaphore ID Queues 0 and 1.

13.14.10 Enhanced Classification Enable Register (Enh_HWC_Ena)

This register enables additional hardware-assisted classification capabilities.

Access Type Read/Write

Base Addresses x'2500 0190'

Reserved																															Enh_IPv4
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Field Name	Bit(s)	Reset	Description
Reserved	31:1		Reserved.
Enh_IPv4	0	0	Enhanced IPv4 classification enable bit. When set to ‘0’, enhanced IPv4 classification is disabled. When set to ‘1’, the following functions are enabled: • A PCT bit for IPv4 forwarding (bit 24 of the PCT) is added. • Three code entry points for IPv4 exception (exception_entry) for three frame formats (Point-to-Point Protocol [PPP], Ethernet with virtual local-area network [VLAN], and Ethernet without VLAN) are added to the HCCIA (see *Section 3.13.1 Hardware Classifier Instruction Address Memory (HCCIA_Memory)* on page 273). The code entry points are located at the current three IPv4 HCCIA entries for the three formats. When Enh_IPV4 is set to ‘1’, an IP frame will go to the exception_entry if: - The PCT entry for IPv4 forwarding is '0'. - The version and header length fields of the IPv4 header are not x'45'. - The first four bits of the Internet Protocol (IP) destination address (DA) are greater than x'E'. • Three code entry points for IPv4 multicast (ip_mc_entry) for three frame formats (PPP, Ethernet with VLAN, and Ethernet without VLAN) are added to the HCCIA table. An IP frame will go to the ip_mc_entry if: - It does not go to the exception entry. - The first four bits of the IP DA equal x'E'. • Three code entry points for IPv4 unicast (uc_entry) for three frame formats (PPP, Ethernet with VLAN, and Ethernet without VLAN) are added to the HCCIA table. An IP frame will go to the uc_entry if: - It does not go to the exception entry. - The first four bits of the IP DA are less than x'E'. • The time to live (TTL) is checked. If TTL ≤ 1, the R1 flags (bit 7) are set.

13.15 Flow Control Structures

13.15.1 Ingress Flow Control Hardware Structures

13.15.1.1 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.

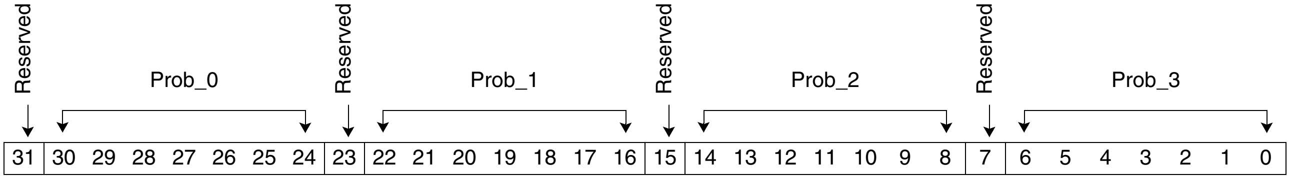
The probability memory occupies 16 entries in the CAB address space. Each probability entry in the CAB is 32 bits wide and contains four 7-bit probabilities. The 4-bit access address to each probability entry comprises two components: the 3-bit QosClass field (QQQ) and the 1-bit Remote Egress Status 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'.

13.15.1.2 Ingress pseudorandom Number Register (I_Rand_Num)

This register contains a 32-bit pseudorandom number used in the Flow Control algorithms. The CAB accesses this register in order to modify its starting point in the pseudorandom sequence. However, a write to this register is not necessary to start the pseudorandom sequence: it starts generating pseudorandom numbers as soon as the reset is finished.

Access Type Read/Write

Base Addresses x'3000 0100'

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

Field Name	Bit(s)	Reset	Description
Rand_Num	31:0		32-bit pseudorandom number.

13.15.1.3 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'

Reserved								FQ_SBFQ_Th								FQ_P0_Th								FQ_P1_Th							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Field Name	Bit(s)	Reset	Description
Reserved	31:24		Reserved.
FQ_SBFQ_Th	23:16	x'FF'	Threshold for comparison to the Ingress Free queue count (BCB_FQ). When the number of buffers indicated by BCB_FQ is less than the number of buffers indicated by FQ_SBFQ_Th, then I_FreeQ_Th I/O is set to '1'.
FQ_P0_Th	15:8	x'FF'	Threshold for comparison to the Ingress Free queue count (BCB_FQ), when determining flow control actions against Priority 0 traffic. When the number of buffers indicated by BCB_FQ is less than the number of buffers indicated by FQ_P0_Th, then flow control hardware discards the frame.
FQ_P1_Th	7:0	x'FF'	Threshold for comparison to the Ingress Free queue count (BCB_FQ), when determining flow control actions against Priority 1 traffic. When the number of buffers indicated by BCB_FQ is less than the number of buffers indicated by FQ_P1_Th, then flow control hardware discards the frame.

13.15.2 Egress Flow Control Structures

13.15.2.1 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 (see *Section 4.7.1 Flow Queues (QCB)* on page 349), and a 1-bit “threshold exceeded” value for the current priority of the combined flow/port queue count (see *Section 4.7.2 Target Port Queues (TPQ)* on page 356).

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.

Reserved	Prob_0							Reserved	Prob_1							Reserved	Prob_2							Reserved	Prob_3						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

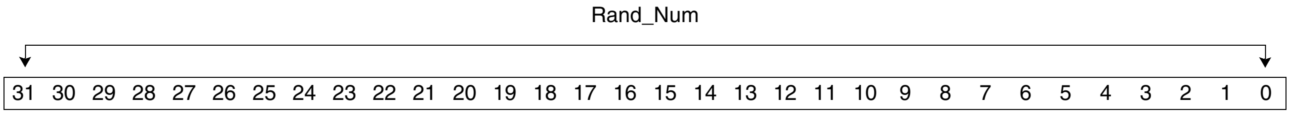
Field Name	Bit(s)	Reset	Description
Reserved	31		Reserved.
Prob_0	30:24		Transmit Probability 0 - transmit probability accessed when ‘FP’ is ‘11’.
Reserved	23		Reserved.
Prob_1	22:16		Transmit Probability 1 - transmit probability accessed when ‘FP’ is ‘10’.
Reserved	15		Reserved.
Prob_2	14:8		Transmit 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’.

13.15.2.2 Egress pseudorandom Number (E_Rand_Num)

This register contains a 32-bit pseudorandom number used in the Flow Control algorithms. The CAB accesses this register in order to modify its starting point in the pseudorandom sequence. However, a write to this register is not necessary to start the pseudorandom sequence: it starts generating pseudorandom numbers as soon as the reset is finished.

Access Type Read/Write

Base Addresses x'B000 0100'



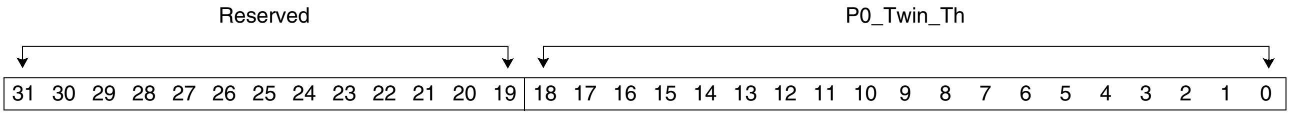
Field Name	Bit(s)	Reset	Description
Rand_Num	31:0		32-bit pseudorandom number.

13.15.2.3 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'



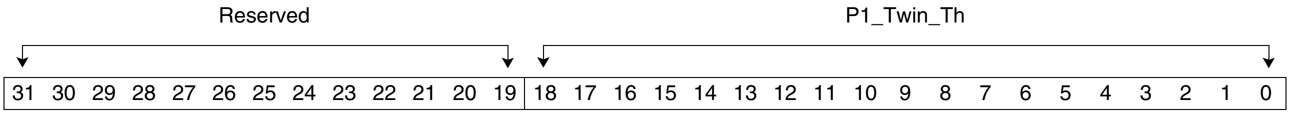
Field Name	Bit(s)	Reset	Description
Reserved	31:19		Reserved.
P0_Twin_Th	18:0	x'0 0000'	P0 Twin Count Threshold value used in the Egress Flow Control Hardware algorithm.

13.15.2.4 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.

13.15.2.5 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 exponentially weighted moving average (EWMA) (see *Section 4.11.2 Egress P0 Twin Count EWMA Register (E_P0_Twin_EWMA)* on page 374). The results of this comparison are placed on the Remote Egress Status Bus.

Access Type Read/Write

Base Addresses x'A040 0400'

Reserved													P0_Twin_EWMA_Th																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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.

13.15.2.6 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(see *Section 4.11.3 Egress P1 Twin Count EWMA Register (E_P1_Twin_EWMA)* on page 375). The results of this comparison are placed on the Remote Egress Status Bus.

Access Type Read/Write

Base Addresses x'A040 0800'

Reserved													P1_Twin_EWMA_Th																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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.

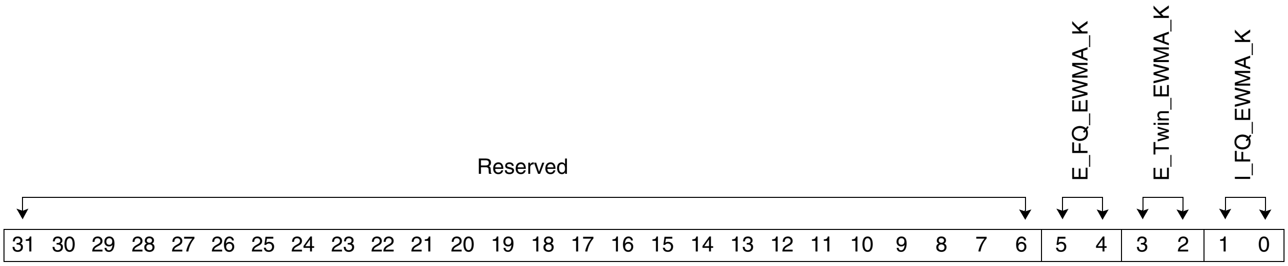
13.15.3 Exponentially Weighted Moving Average Constant (K) Register (EWMA_K)

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’



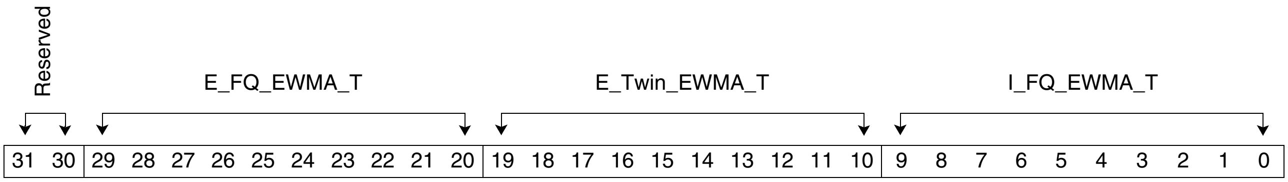
Field Name	Bit(s)	Reset	Description
Reserved	31:6		Reserved.
E_FQ_EWMA_K	5:4	00	K value for the Egress Free Queue Count Exponentially Weighted Moving Average calculation in the Egress Flow Control Hardware.
E_Twin_EWMA_K	3:2	00	K value for the Egress P0/P1 Twin Count EWMA calculation in the Egress Flow 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.

13.15.4 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 nonzero.

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.

13.15.5 Flow Control Force Discard Register (FC_Force_Discard)

This register is used to send egress packets to the discard port when the scheduler is disabled, or the designated discard queue control block (QCB) when the scheduler is enabled. This register is added for debug purposes, but may be useful for soft reset/soft restart functions.

Access Type Read/Write

Base Addresses x'A040 2000'

Base Address Offset 0

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

Base Address Offset 1

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

Base Address Offset 0

Field Name	Bit(s)	Reset	Description
Discard_tp	31:30, 27:26, 23:22, 19:18, 15:14, 11:10, 7:6, 3:2, 0	x'0 0000'	Each bit of the Discard_tp field corresponds a port identifier value (see *Table 9-1: Target Port Mapping* on page 499). When a bit is set to '1', and the FCInfo field is not set to x'F' during an enqueue, then the enqueued egress frame for the port identifier is discarded by the flow control logic.
Reserved	29:28, 25:24, 21:20, 17:16, 13:12, 9:8, 5:4, 1		Reserved.

Base Address Offset 1

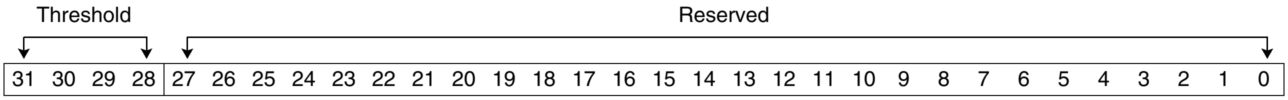
Field Name	Bit(s)	Reset	Description
Reserved	31:8, 5:4, 1:0		Reserved.
Discard_tp(39:38, 35:34)	7:6, 3:2	0	Each bit of the Discard_tp field corresponds a port identifier value(see *Table 9-1: Target Port Mapping* on page 499). When a bit is set to '1', and the FCInfo field is not set to x'F' during an enqueue, then the enqueued egress frame for the port identifier is discarded by the flow control logic.

13.16 Egress CDM Stack Threshold Register (E_CDM_Stack_Th)

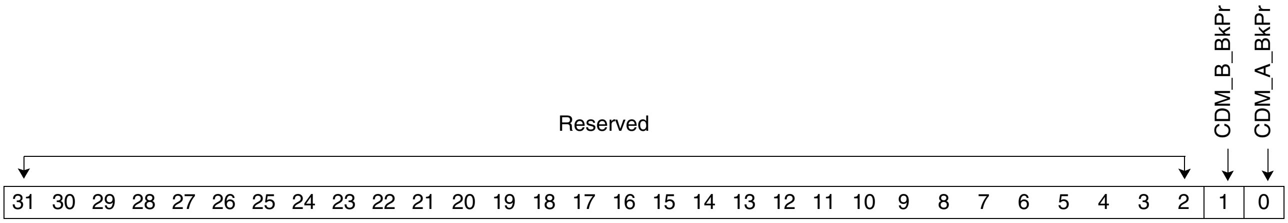
Access Type Read/Write

Base Address x‘A000 1800’

Base Address Offset 0



Base Address Offset 1



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
Threshold	31:28	x‘8’	E-CDM Stack Threshold value. When this threshold is violated (threshold value is greater than or equal to the count of empty entries in the E-CDM Stack), send grant is set to its disable state. The depth of the CDM Stack is 32 entries, and this field has a resolution of two buffers (i.e., a value of x'8' indicates 16 entries).
Reserved	27:0		Reserved.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
Reserved	31:2		Reserved.
CDM_BkPr	0	0	CDM force backpressure control. When set to ‘1’, the Send_Grant signal is set to ‘0’, indicating that the interface is unable to accept additional data cells. When set to ‘0’, Send_Grant is set based on a E_CDM_Stack Threshold violation.

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’

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

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 - 11/10). The value in this register can be modified; however, the new value cannot 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 NP2G'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'

FQ_ES_Thresh_0															Reserved																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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 IEW is discarded (the number of frames discarded is counted). • Frames that have started reassembly 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’

FQ_ES_Thresh_1															Reserved																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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’

FQ_ES_Thresh_2															Reserved																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

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 Egress Frame Data Queue Thresholds (E_GRx_GBx_th)

The NP2G supports four queues for egress data frames GR0 and GR1 (high priority) and GB0 and GB1 (low priority). See *Section 4.5.2 General Queues (GR0, GR1, GB0, GB1, GFQ, GPQ, GTQ)* on page 328 for more information.

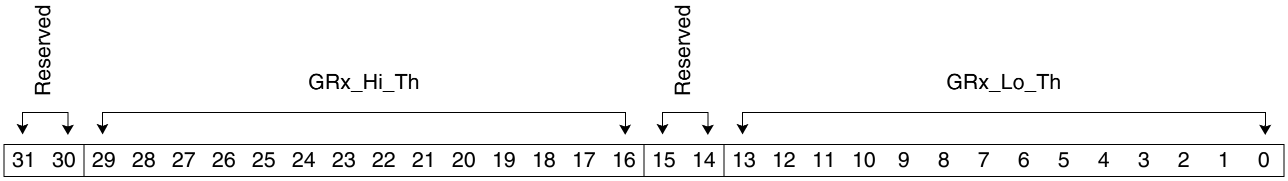
Thresholds are provided for the egress frame data queues to allow configuration of discard actions when these queues become full. When a threshold is violated, the frame attempting to be enqueued in the general queue is sent to the Egress General Discard Queue (E-GDQ) instead (see *Section 4.5.3 Discard Queue (GDQ)* on page 331). When the E-GDQ is full, no further cells are pulled from the IEW until the condition is resolved.

Each threshold has a 32-entry resolution; high and low thresholds are defined for each egress frame data queue. When exceeding the high threshold, the above discard actions occur and continue until the number of entries in the queue drops below the low threshold.

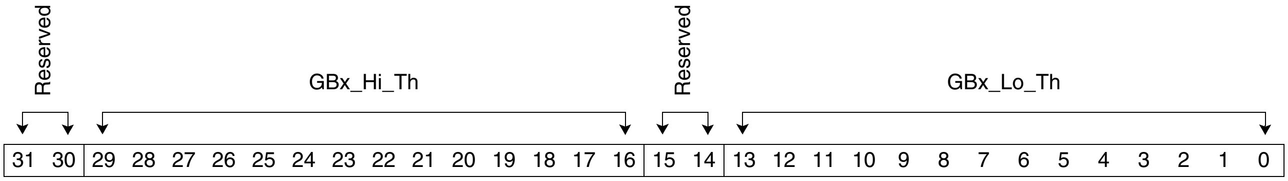
Access Type Read/Write

Base Address x‘A000 1200’

Base Address Offset 0



Base Address Offset 1



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
Reserved	31:30		Reserved.
GRx_Hi_Th	29:16	x‘3FFF’	GRx High Threshold. When a GRx queue (GR0 or GR1) exceeds this threshold, new frames are enqueued to the discard stack until the number of entries drops below the GRx low threshold.
Reserved	15:14		Reserved.
GRx_Lo_Th	13:0	x‘3FFF’	GRx Low Threshold. If discards are occurring because the GRx high threshold was exceeded, the number of entries in the corresponding GRx queue (GR0 or GR1) must drop below this threshold before enqueues to the corresponding GRx queue can resume.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
Reserved	31:30		Reserved.
GBx_Hi_Th	29:16	x‘3FFF’	GBx High Threshold. When a GBx queue (GB0 or GB1) exceeds this threshold, new frames are enqueued to the discard stack until the number of entries drops below the GBx low threshold.
Reserved	15:14		Reserved.
GBx_Lo_Th	13:0	x‘3FFF’	GBx Low Threshold. If discards are occurring because the GBx high threshold was exceeded, the number of entries in the corresponding GBx queue (GB0 or GB1) must drop below this threshold before enqueues to the corresponding GBx queue can resume.

13.20 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 queue control block (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 *Section 6* beginning on page 141 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’

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

Field Name	Bit(s)	Reset	Description
Reserved	31:11		Reserved.
Discard_QID	10:0	x‘029’	The Discard Queue Identification (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.

13.21 Bandwidth Allocation Register (BW_Alloc_Reg)

The bandwidth (BW) allocation register is used to assure bandwidth to the egress data store for the EPC Data Store Coprocessor and the Dispatch Unit. For the EPC Data Store Coprocessor, this register provides assured bandwidth when writing to the egress data store. For the EPC dispatch unit, it provides assured bandwidth when reading the egress data store.

Access Type Read/Write

Base Addresses x‘A000 2800’

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

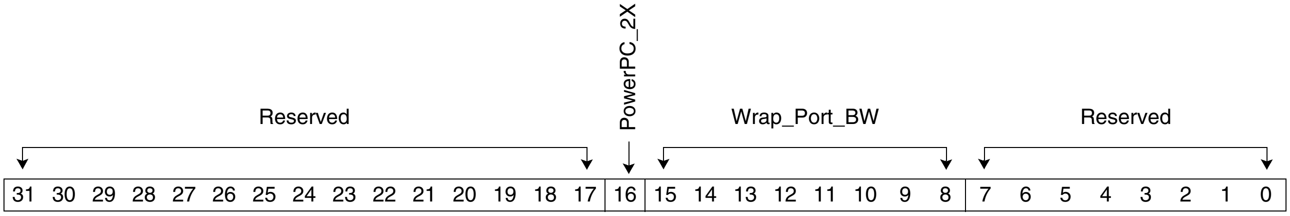
Field Name	Bit(s)	Reset	Description
Reserved	31:16		Reserved.
Disp_BW_Alloc	15:8	0	Dispatch Unit read BW allocation control value. An 8-bit countdown value. A value of 0 disables this function. The value is loaded into a control registers and is decremented each data store access window. While the control register is nonzero, the following priority is observed for read access to the egress data stores: 8/9ths of the time 1/9th of the time 1. PMM - DMU A-D read requests PMM - DMU A, C, D read requests 2. Discard Port Dispatch Unit Dispatch Unit EPC Data Store Coprocessor read 3. EPC Data Store Coprocessor read Discard Port 4. PMM - Wrap DMU PMM - Wrap DMU Once the control register decrements to zero, the following priority is observed: 8/9th of the time 1/9th of the time 1. Dispatch Unit Dispatch Unit 2. PMM - DMU A-D read requests PMM - DMU A, C, D read requests Discard Port EPC Data Store Coprocessor read 3. EPC Data Store Coprocessor read Discard Port 4. PMM - Wrap DMU PMM - Wrap DMU When the Dispatch Unit is serviced, or a DRAM refresh occurs for DS0/DS1, the control register is reloaded with the configuration value.
DS_BW_Alloc	7:0	0	Data store coprocessor write BW allocation control value. An 8-bit countdown value. A value of 0 disables this function. This value is loaded into a control register and is decremented each data store access window. While the control register is nonzero, the IEW interface has high priority for data store write access. The EPC data store coprocessor write requests must wait until there is a free window. Once the control register has counted down to zero, the EPC request for the data store write access window has high priority. The control register remains at zero until an EPC data store coprocessor write request is serviced. Whenever an EPC data store write window request is serviced and the configuration register is nonzero, the control register is reloaded with the configuration value.

13.22 Miscellaneous Controls Register (MISC_CNTRL)

This register sets the speed of the embedded PowerPC processor and sets a guaranteed egress data store bandwidth for the wrap port

Access Type Read/Write

Base Addresses x‘9001 6E00’



Field Name	Bit(s)	Reset	Description
Reserved	31:17		Reserved.
PowerPC_2X	16	0	PowerPC processor speed. 0 PowerPC processor runs at 133 MHz. 1 PowerPC processor runs at 266 MHz.
Wrap_Port_BW	15:8	x’00’	Wrap port read bandwidth allocation control value. An 8-bit countdown value. A value of 0 disables this function. The value is loaded into a control register and is decremented at each data store access window. When the control register is nonzero, the priority for read access to the egress data store is as described in the Bandwidth Allocation register (see *Section 13.21* on page 482). When the control register becomes zero, the priorities shown in the table for the bandwidth allocation register change so that the wrap port is prioritized above the dispatch unit in all four cases.
Reserved	7:0		Reserved.

13.23 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. This register must be set prior to initialization (see *Initialization Register (Init)* [page 451]) to affect the FCB FQ size. Any changes after initialization will not affect the number of available FCBs.

Access Type Read/Write

Base Address x'A000 2200'

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

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 Reserved
Reserved	29:0		Reserved.

Note: The following settings are recommended based on the size of the D4 DRAM which is configured in bits 2:1 of the *DRAM Parameter Register (DRAM_Parm)* (page 434).

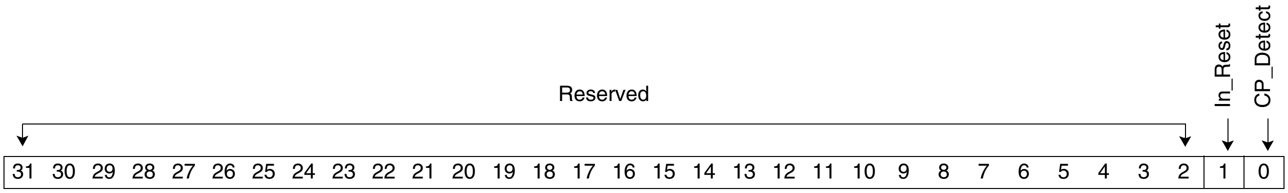
DRAM_Size	FCB_FQ_Max
00	01
01	10
10	10
11	Reserved

13.24 Data Mover Unit (DMU) Configuration Registers

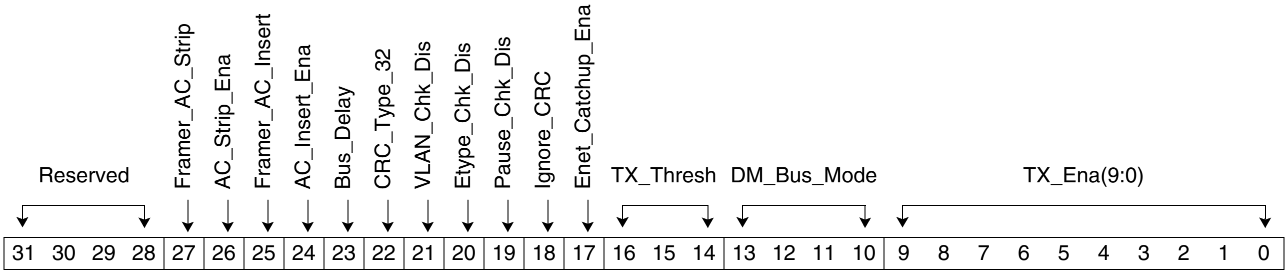
There are three Data Mover Units (DMU) configured for internal medium access control (MAC) operation, for external connection detection (i.e., attached control point detection), and for external bus operation (i.e., Gigabit Media-Independent Interface [GMII], Serial Media-Independent Interface [SMII], Ten-Bit Interface [TBI], and packet-over-SONET [POS] framer).

Access Type	Base Address Offset 0	Read Only
	Base Address Offset 1 Base Address Offset 2 Base Address Offset 3	Read/Write
Base Address	DMU_A	x'A001 0010'
	DMU_C	x'A001 0040'
	DMU_D	x'A001 0080'

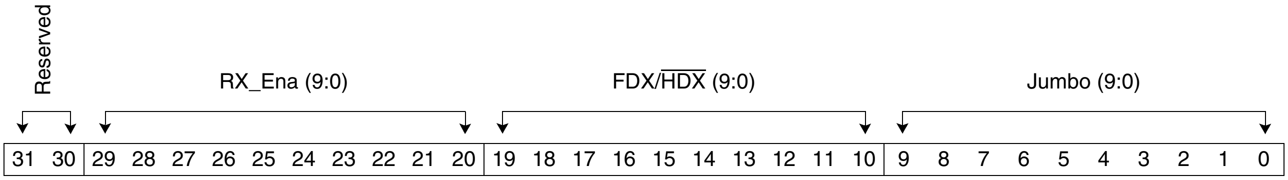
Base Address Offset 0



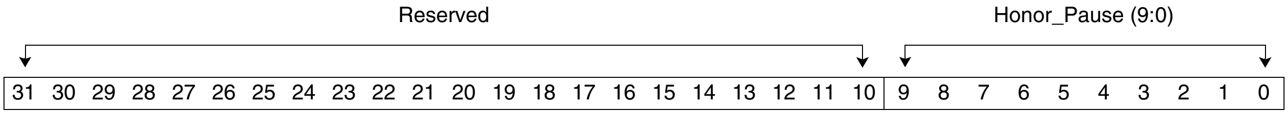
Base Address Offset 1



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 Physical MAC Multiplexer (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 address control (AC) field to the network processor. 1 Framer does not pass AC field to the network processor. The cyclic redundancy check (CRC) performed is modified to account for the missing field. An 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 POS MAC. This field does not control CRC generation for the Ethernet MAC. The Ethernet MAC can generate only 32-bit CRC. 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 Range available for all DMU_Bus_Mode settings 101-111 Range available only for Gigabit Ethernet, POS OC-3, and POS OC-12 DMU_Bus_Mode settings The normal setting for this field is half of the available buffers unless toggle mode is used, in which case this field should be set to ‘100’ for all modes. 011 Fast Ethernet mode 100 All other modes
DM_Bus_Mode	13:10	1010	Data Mover Bus Mode configures the mode in which the data mover bus operates: 0000 Reserved 0001 10/100 Ethernet SMII Mode (DMU C & D only) 0010 Gigabit Ethernet GMII Mode 0011 Gigabit Ethernet TBI Mode 0100 POS OC-12 Mode; nonpolling POS support 0101 POS 4xOC-3 mode; polling POS support (DMU C & D only) 0110-1001 Reserved 1010 CP Detect Mode 1011 Debug Mode (DMU D only). This mode is supported only when its use is directed by an IBM Network Processor Applications Engineer. 1100-1101 Reserved 1110 DMU Disabled 1111 Reserved **Note:** When configuring the DMU, the DMU_Bus_Mode field must be set first. Subsequent CAB writes can used to configure the remaining fields.
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

Field Name	Bit(s)	Reset	Description
FDX/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
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 The maximum length of a jumbo frame is determined by the value in the *Ethernet Jumbo Frame Size Register (EN_Jumbo_FS)* (page 489).

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

13.25 Frame Pad Configuration Register (DMU_Pad)

This register configures the hardware-assisted frame alteration padding function for each DMU. The minimum number of bytes per frame and the padding byte are specified. The padding function is available only when the DM_Bus_Mode field of the *Data Mover Unit (DMU) Configuration Register* (page 485) is set to ‘0001’, ‘0010’, or ‘0011’ (one of the Ethernet MAC modes of operation).

Access Type	Read/Write	
Base Address	DMU_A_Pad	x‘A002 0100’
	DMU_C_Pad	x‘A002 0400’
	DMU_D_Pad	x‘A002 0800’

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

Field Name	Bit(s)	Reset	Description
Reserved	31:16		Reserved.
Pad_Byte	15:8	x‘00’	Contains the value used to pad the frame to minimum size.
Min_Size	7:0	0	The minimum size value (in bytes). When set to zero, the padding function is disabled.

13.26 Ethernet Jumbo Frame Size Register (EN_Jumbo_FS)

This register controls the maximum frame size supported by the network processor when the DMU is configured for Ethernet operation. This register configuration is used when jumbo frame support is enabled in the *Data Mover Unit (DMU) Configuration Register* (page 485). The network processor is constrained to a maximum of 14K (14336) bytes. Frames received by the network processor that exceed the length specified by this register are aborted during reception. Note that this value is modified by an additional four bytes when a VLAN-tagged frame is detected.

Access Type	Read/Write
Base Address	x‘A040 8000’

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

Field Name	Bit(s)	Reset	Description
Reserved	31:14		Reserved.
Jumbo_Max	13:0	x‘233A’	Defines the maximum length for jumbo Ethernet frames. This value is used to determine when to abort receipt of a frame and/or increment long frame counts. The value used to cause the above actions is increased by four bytes when a VLAN-tagged frame is detected.

13.27 QD Accuracy Register (QD_Acc)

The *QD Accuracy Register (QD_Acc)* tunes the egress scheduler's weighted fair queueing (WFQ) rings (see *Section 6.4 The Egress Scheduler* on page 152*).* The values assure fairness and some benefit to queues with lower defined QD values that 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, C, D).

Access Type Read/Write

Base Address x'A002 4000'

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

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.
Reserved	23:20	0	Reserved.
QD_Acc_DMU_A	19:16	0	QD Accuracy value used for DMU_A.
Reserved	15:0		Reserved.

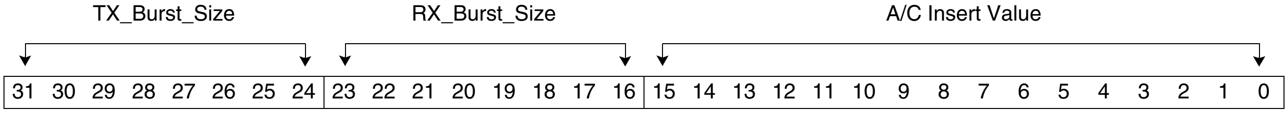
13.28 Packet Over SONET Control Register (POS_Ctrl)

One configuration register per DMU is provided to control POS framer interaction. This register configures transmit and receive burst sizes and sets the value used for AC field insertion.

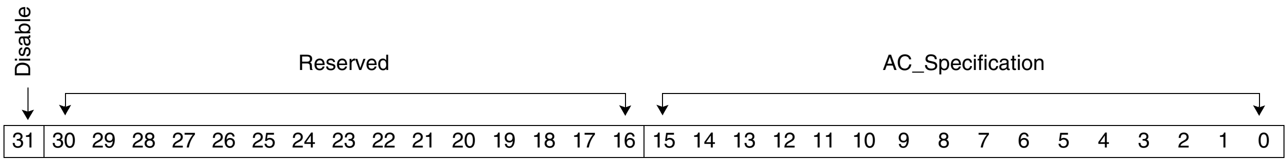
The AC_Specification value allows per-port configuration of the AC value used when the AC_Insert_Ena, Framer_AC_Strip, and/or Framer_AC_Insert fields are set to ‘1’. When either the Framer_AC_Strip or Framer_AC_Insert field is set to ‘1’, an initial CRC value (determined by the AC value assumed for the port) must be loaded into offsets 12-15. For cases where both the Framer_AC_Strip and Framer_AC_Insert fields are set to ‘1’, the AC byte assumed for the port on ingress (Framer_AC_Strip) and egress (Framer_AC_insert) must be the same value. A backward compatibility mode allows a single insert value to be specified for all ports (offset 0) on the transmit side and a value of x'FF03' to be assumed for all ports on the receive side.

Access Type	Read/Write	
Base Address	DMU_A	x‘A004 0100’
	DMU_C	x‘A004 0400’
	DMU_D	x‘A004 0800’

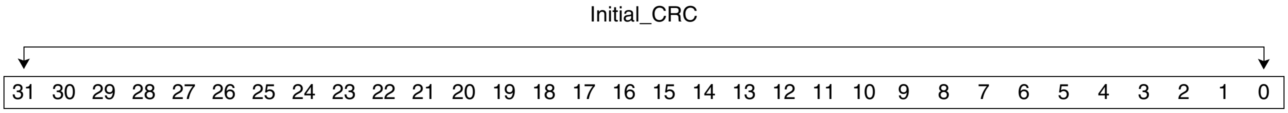
Base Address Offset 0



Base Address Offset 8, Port 0
Base Address Offset 9, Port 1 (for DMU C and D only)
Base Address Offset 10, Port 2 (for DMU C and D only)
Base Address Offset 11, Port 3 (for DMU C and D only)



Base Address Offset 12, Port 0
Base Address Offset 13, Port 1 (for DMU C and D only)
Base Address Offset 14, Port 2 (for DMU C and D only)
Base Address Offset 15, Port 3 (for DMU C and D only)



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
TX_Burst_Size	31:24	x'10'	Transmit burst size. Used only for OC-3 modes of operation. OC-12 interfaces transmit until the framer de-asserts TxPFA. 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’.

Base Address Offset 8, Port 0
Base Address Offset 9, Port 1 (for DMU C and D only)
Base Address Offset 10, Port 2 (for DMU C and D only)
Base Address Offset 11, Port 3 (for DMU C and D only)

Field Name	Bit(s)	Reset	Description
Disable	31	1	Disables the per-port AC byte specification for port *N.* 0 When the AC_Insert_En field in the *Data Mover Unit (DMU) Configuration Register* (page 485) is set to ‘1’, the AC_Specification field is used. When the Framer_AC_Strip field in the *DMU Configuration Register* is set to ‘1’, the Initial_CRC value (this offset plus four) plus the frame data are used for CRC checking. When the Framer_AC_Insert field in the *DMU Configuration Register* is set to ‘1’, the Initial_CRC value (this offset plus four) plus the frame data are used for CRC generation. 1 When the AC_Insert_En field for the *DMU Configuration Register* is set to ‘1’, the A/C Insert Value found in offset 0 is used. When the Framer_AC_Strip field in the *DMU Configuration Register* is set to ‘1’, an AC value of x‘FF03’ plus the frame data are used for CRC checking. When the Framer_AC_insert field of the *DMU Configuration Register* is set to ‘1’, an AC value of x‘FF03’ plus the frame data are used for CRC generation.
Reserved	30:16		Reserved.
AC_Specification	15:0	x‘FF03’	Used when the AC_Insert_En field of the *DMU Configuration Register* is set to ‘1’, and the Disable field (bit 31 above) is set to ‘0’.

Base Address Offset 12, Port 0
Base Address Offset 13, Port 1 (for DMU C and D only)
Base Address Offset 14, Port 2 (for DMU C and D only)
Base Address Offset 15, Port 3 (for DMU C and D only)

Field Name	Bit(s)	Reset	Description
Initial_CRC	31:0	x‘1382 D02D’	Initial CRC value for Port *N*. Used for CRC generation (when the Framer_AC_Insert field in the *DMU Configuration Register* is set to ‘1’) or checking (when the Framer_AC_Strip field in the *DMU Configuration Register* is set to ‘1’) when the Disable field (this offset minus four) is set to ‘0’. The reset value is based on an AC value of x‘FF03’ and the 32-bit CRC.

13.28.1 Initial CRC Value Determination

The Tcl script below illustrates the algorithm for determining the Initial_CRC value when the user provides the software application program interface (API) with the AC byte specification when the Framer_AC_Strip and/or Frame_AC_Insert fields of the *Data Mover Unit (DMU) Configuration Register* (page 485) are set to ‘1’.

```
# Initial CRC value generation for POS AC specifications
proc initcrc { ACBytes CRCType } {
# ACBytes is a bit vector passed in hex format (eg 0xAACC)
# CRCType is either 32 or 16
# Golden CRC test:  initcrc 0xFF03 32  results in 0x1382D02D
#                   initcrc 0xFF03 16  results in 0x0000C7BC
# These values are set into the configuration register as shown. ie CABWRITE $address 0x0000C7BC
#       puts "Setting up initial next_crc and crc values"
        for {set i 0} {$i < 32} {incr i} {
                set next_crc($i) 1              ;# initialize next_CRC value 0 up to 31
                set crc($i) $next_crc($i)       ;# initialize CRC value  0 up to 31
        }
#       puts "Creating bit array of input ACbytes"
        for {set i 0} {$i < 16} {incr i} {
                set temp [expr { ($ACBytes >> $i) & 0x0001 }]
                set input_data_stored($i) $temp         ;# input data 15 down to 0
#               puts [format "input_data_stored bit %u : %u" $i $input_data_stored($i)]
        }
        for {set pass 0} {$pass < 2} {incr pass} {
#           puts [format "Starting pass %u" $pass]
# set up input data for CRC generation.  Data is 7 down to 0
            for {set j 0} {$j < 8} {incr j} {
              set data_stored($j) $input_data_stored([expr {$j + (1-$pass) * 8}])
            }
            if {($CRCType == 32)} {
#               puts "Generating 32 bit CRC"
                for {set i 0} {$i < 8} {incr i}  {
                    set crcbit [expr {31 - $i}]
#                   puts [format "datastored(%u): %u \t crc(%u): %u" $i $data_stored($i) $crcbit
                         $crc($crcbit)]
                    set n($i) [expr {$data_stored($i) ^ $crc($crcbit)}]
                }
                set next_crc(31) [expr {$crc(23) ^ $n(2)}]
                set next_crc(30) [expr {$crc(22) ^ $n(0) ^ $n(3)}]
                set next_crc(29) [expr {$crc(21) ^ $n(0) ^ $n(1) ^ $n(4)}]
                set next_crc(28) [expr {$crc(20) ^ $n(1) ^ $n(2) ^ $n(5)}]
                set next_crc(27) [expr {$crc(19) ^ $n(0) ^ $n(2) ^ $n(3) ^ $n(6)}]
                set next_crc(26) [expr {$crc(18) ^ $n(1) ^ $n(3) ^ $n(4) ^ $n(7)}]
                set next_crc(25) [expr {$crc(17) ^ $n(4) ^ $n(5)}]
                set next_crc(24) [expr {$crc(16) ^ $n(0) ^ $n(5) ^ $n(6)}]
                set next_crc(23) [expr {$crc(15) ^ $n(1) ^ $n(6) ^ $n(7)}]
                set next_crc(22) [expr {$crc(14) ^ $n(7)}]
                set next_crc(21) [expr {$crc(13) ^ $n(2)}]
                set next_crc(20) [expr {$crc(12) ^ $n(3)}]
                set next_crc(19) [expr {$crc(11) ^ $n(0) ^ $n(4)}]
                set next_crc(18) [expr {$crc(10) ^ $n(0) ^ $n(1) ^ $n(5)}]
                set next_crc(17) [expr {$crc(9) ^ $n(1) ^ $n(2) ^ $n(6)}]
                set next_crc(16) [expr {$crc(8) ^ $n(2) ^ $n(3) ^ $n(7)}]
                set next_crc(15) [expr {$crc(7) ^ $n(0) ^ $n(2) ^ $n(3) ^ $n(4)}]
                set next_crc(14) [expr {$crc(6) ^ $n(0) ^ $n(1) ^ $n(3) ^ $n(4) ^ $n(5)}]
                set next_crc(13) [expr {$crc(5) ^ $n(0) ^ $n(1) ^ $n(2) ^ $n(4) ^ $n(5) ^ $n(6)}]
                set next_crc(12) [expr {$crc(4) ^ $n(1) ^ $n(2) ^ $n(3) ^ $n(5) ^ $n(6) ^ $n(7)}]
                set next_crc(11) [expr {$crc(3) ^ $n(3) ^ $n(4) ^ $n(6) ^ $n(7)}]
                set next_crc(10) [expr {$crc(2) ^ $n(2) ^ $n(4) ^ $n(5) ^ $n(7)}]
                set next_crc(9) [expr {$crc(1) ^ $n(2) ^ $n(3) ^ $n(5) ^ $n(6)}]
                set next_crc(8) [expr {$crc(0) ^ $n(3) ^ $n(4) ^ $n(6) ^ $n(7)}]
                set next_crc(7) [expr {$n(0) ^ $n(2) ^ $n(4) ^ $n(5) ^ $n(7)}]
                set next_crc(6) [expr {$n(0) ^ $n(1) ^ $n(2) ^ $n(3) ^ $n(5) ^ $n(6)}]
                set next_crc(5) [expr {$n(0) ^ $n(1) ^ $n(2) ^ $n(3) ^ $n(4) ^ $n(6) ^ $n(7)}]
                set next_crc(4) [expr {$n(1) ^ $n(3) ^ $n(4) ^ $n(5) ^ $n(7)}]
                set next_crc(3) [expr {$n(0) ^ $n(4) ^ $n(5) ^ $n(6)}]
                set next_crc(2) [expr {$n(0) ^ $n(1) ^ $n(5) ^ $n(6) ^ $n(7)}]
                set next_crc(1) [expr {$n(0) ^ $n(1) ^ $n(6) ^ $n(7)}]
                set next_crc(0) [expr {$n(1) ^ $n(7)}]
                } else {
# 16 bit CRC
#               puts "Generating 16 bit crc"
                for {set i 0} {$i < 8} {incr i}  {
                      set n($i) [expr {$data_stored($i) ^ $crc([expr {15 - $i}])}]
                }
                for {set i 16} {$i < 32} {incr i} {
                    set next_crc($i) 0
                }
                set next_crc(15) [expr {$n(0) ^ $n(4) ^ $crc(7)}]
                set next_crc(14) [expr {$n(1) ^ $n(5) ^ $crc(6)}]
                set next_crc(13) [expr {$n(2) ^ $n(6) ^ $crc(5)}]
                set next_crc(12) [expr {$n(0) ^ $n(3) ^ $n(7) ^ $crc(4)}]
                set next_crc(11) [expr {$n(1) ^ $crc(3)}]
                set next_crc(10) [expr {$n(2) ^ $crc(2)}]
                set next_crc(9)  [expr {$n(3) ^ $crc(1)}]
                set next_crc(8)  [expr {$n(0) ^ $n(4) ^ $crc(0)}]
                set next_crc(7)  [expr {$n(0) ^ $n(1) ^ $n(5)}]
                set next_crc(6)  [expr {$n(1) ^ $n(2) ^ $n(6)}]
                set next_crc(5)  [expr {$n(2) ^ $n(3) ^ $n(7)}]
                set next_crc(4)  [expr {$n(3)}]
                set next_crc(3)  [expr {$n(0) ^ $n(4)}]
                set next_crc(2)  [expr {$n(1) ^ $n(5)}]
                set next_crc(1)  [expr {$n(2) ^ $n(6)}]
                set next_crc(0)  [expr {$n(3) ^ $n(7)}]
                } ; # if crctype
# Move Next_crc into CRC
           for {set i 0} {$i < 32} {incr i} {
               set crc($i) $next_crc($i)       ;# CRC value  0 up to 31
           }
        } ; # for pass
        set golden_crc 0x00000000
        for {set i 0} {$i < 32} {incr i} {
#           puts [format "%u : %u" $i $crc($i)]
            set golden_crc [expr {($golden_crc << 1) | $crc([expr {31 - $i}])}]
        }
        puts [format "Golden CRC is %#010x" $golden_crc]
}
```

13.29 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’

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

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.

13.30 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 Destination Address (DA), Source Address (SA), and Type fields of the frame are stripped and the frame is queued onto the Guided Frame Queue (GFQ).

Access Type Read/Write

Base Addresses x'A001 1000'

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

Field Name	Bit(s)	Reset	Description
E_Type	31:16	x'0000'	Ethernet Type used for encapsulated guided traffic.
Reserved	15:0		Reserved.

13.31 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 General Data Queue (GDQ).

Access Type Read/Write

Base Addresses x'A001 2000'

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

Field Name	Bit(s)	Reset	Description
E_Type	31:16	x'0000'	Ethernet Type used for encapsulated control point (CP) data traffic.
Reserved	15:0		Reserved.

13.32 Source Address Array (SA_Array)

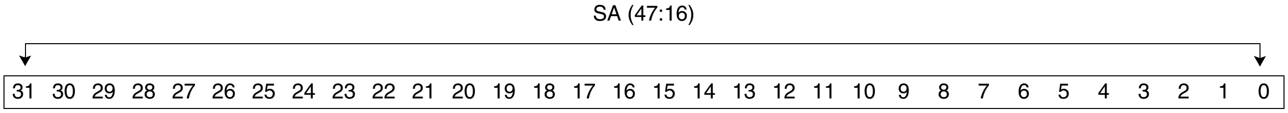
The SA Array is a register array containing 64 Source Address values. The SA Pointer from the egress FCB references elements of this register array. The value retrieved from the SA Array is used to insert or overlay the SA field of transmitted frames during egress frame alteration.

Access Type Read/Write

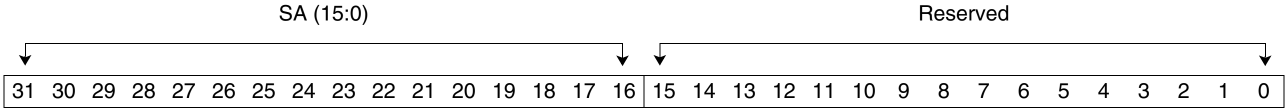
Base Addresses **Note:** Each DMU listed below contains 64 entries. Nibbles 5 and 6 of the base address are incremented by x‘01’ for each successive entry, represented by “##,” and ranging from x‘00’ to x‘3F’. The word offset is represented by “W,” and ranges from x‘0’ to x‘1’.

DMU_A	x‘8810 0##W’
DMU_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 from the SAPtr field 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 from the SAPtr field of the FCB.
Reserved	15:0		Reserved.

13.33 Destination Address Array (DA_Array)

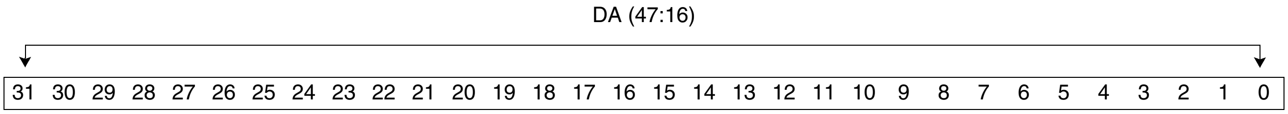
The DA Array is a register array containing 64 Destination Address values. The DAPtr field from the egress FCB, when enabled by the egress FCB field EN_HWA(2), is used to reference elements of this register array. The value retrieved from the DA Array is used to insert or overlay the DA field of transmitted frames during egress frame alteration.

Access Type Read/Write

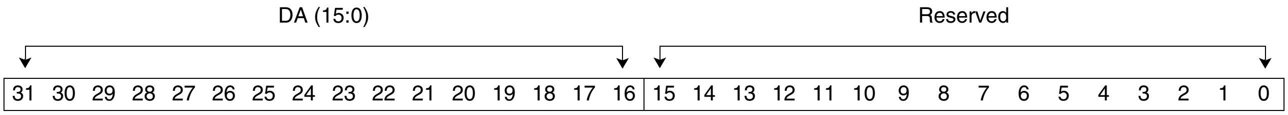
Base Addresses **Note:** Each DMU listed below contains 64 entries. Nibbles 5 and 6 of the base address are incremented by x‘01’ for each successive entry, represented by “##,” and ranging from x‘00’ to x‘3F’. The word offset is represented by “W,” and ranges from x‘0’ to x‘1’.

DMU_A	x‘8830 0##W’
DMU_C	x‘8832 0##W’
DMU_D	x‘8833 0##W’
Wrap	x‘8834 0##W’

Base Address Offset 0



Base Address Offset 1



Base Address Offset 0

Field Name	Bit(s)	Reset	Description
DA(47:16)	31:0	Not defined	Destination Address value. When enabled by FCB field EN_HWA(2), the DMU accesses a DA value when it performs SA/DA insert/overlay egress frame alteration functions. The data is the destination address that is either overlaid or inserted into a frame. The entry address is from the FCB DAPtr field.

Base Address Offset 1

Field Name	Bit(s)	Reset	Description
DA(15:0)	31:16	Not defined	Destination Address value. When enabled by FCB field EN_HWA(2), the DMU accesses a DA value when it performs SA/DA insert/overlay egress frame alteration functions. The data is the destination address that is either overlaid or inserted into a frame. The entry address is from the FCB DAPtr field.
Reserved	15:0		Reserved.

13.34 Programmable I/O Register (PIO_Reg)

This configuration register provides the control for the programmable input and output (PIO) signal pins (see *Section 2.1.8 Miscellaneous Pins* on page 75).

Access Type Read/Write

Base Address x‘A040 4000’

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

Field Name	Bit(s)	Reset	Description
Reserved	30:9		Reserved.
PIO_state	8:6		Current value on the PIO(2:0) signal pins. PIO signal pin mapping to register bits is: bit PIO 8 PIO(2) 7 PIO(1) 6 PIO(0)
PIO_enable	5:3	0	Controls PIO(2:0) driver state. Control values are: 0 Driver is in tristate 1 Driver is enabled. PIO signal pin mapping to register bits is: bit PIO 5 PIO(2) 4 PIO(1) 3 PIO(0)
PIO_write	2:0	0	Value to be driven onto PIO(2:0) signal pins when the corresponding driver is enabled. PIO signal pin mapping to register bits is: bit PIO 2 PIO(2) 1 PIO(1) 0 PIO(0)

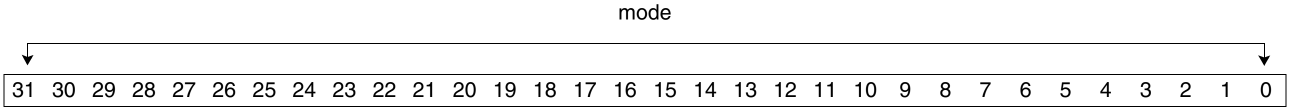
13.35 Ingress-to-Egress Wrap Configuration Registers

These registers define the operating parameters for the IEW function.

13.35.1 IEW Configuration 1 Register (IEW_Config1)

Access Type Read/Write

Base Address x‘A000 0110’

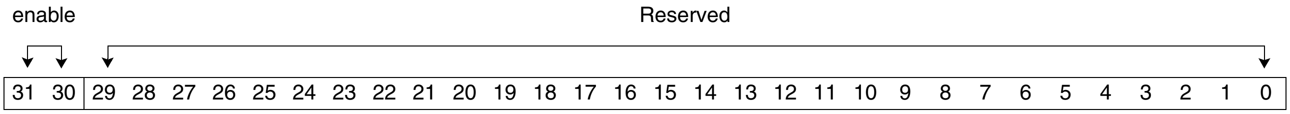


Field Name	Bit(s)	Reset	Description
mode	31:0	x'2003 1000'	Set the mode to x'A003 2000' for initialization. After initialization is complete (see *Section 13.7.2 Initialization Done Register (Init_Done)* on page 452), set the mode to x'A00B 2000', which is the operational state.

13.35.2 IEW Configuration 2 Register (IEW_Config2)

Access Type Read/Write

Base Address x‘A002 0080’

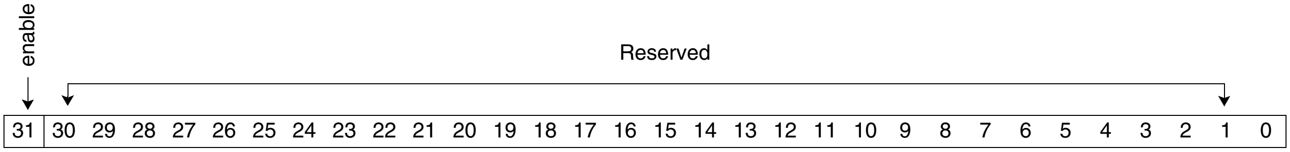


Field Name	Bit(s)	Reset	Description
enable	31:30	‘00’	00 Disabled 11 Enabled
Reserved	29:0		Reserved.

13.35.3 IEW Initialization Register (IEW_Init)

Access Type Read/Write

Base Address x‘A000 0210’



Field Name	Bit(s)	Reset	Description
enable	31	0	Setting this field to ‘1’ begins the IEW initialization.
Reserved	30:0		Reserved.

14. Electrical and Thermal Specifications

The NP2G utilizes IBM CMOS SA-27E technology.

Table 14-1. 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	-40 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 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 1 of 18)

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	$\overline{\text{IEW\_Clk\_B}}$	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 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 2 of 18)

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	$\overline{\text{DS0\_CS}}$	11.81
A32	DS0_Data(01)	10.8
A33	DS0_Data(03)	11.7
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
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
AA32	DMU_A(21)	9.55
AA33	DMU_A(20)	9.95
AB05	LU_Addr(14)	7.81
AB07	LU_Addr(03)	7.51

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 3 of 18)

Grid Position	Signal Name	Total InCap
AB09	LU_Addr(16)	6.91
AB11	D0_Data(03)	4.7
AB13	D0_Data(28)	5
AB15	D0_Addr(05)	6.11
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
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
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
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
AD07	LU_Addr(12)	7.71

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 4 of 18)

Grid Position	Signal Name	Total InCap
AD09	D0_Data(01)	5.9
AD11	D0_Data(23)	5.2
AD13	DB_BA(1)	6.5
AD19	$\overline{\mathrm{D3\_WE}}$	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
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
AE13	$\overline{\mathrm{DB\_RAS}}$	6.7
AE15	D3_Data(03)	5.8
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
AF07	LU_Addr(13)	7.91

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 5 of 18)

Grid Position	Signal Name	Total InCap
AF09	D0_Data(20)	6.5
AF11	D0_Addr(12)	7.51
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
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
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	$\overline{\text{DA\_RAS}}$	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

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 6 of 18)

Grid Position	Signal Name	Total InCap
AG30	PCI_AD(21)	10.95
AG31	PCI_AD(20)	11.05
AG32	PCI_AD(19)	11.75
AG33	PCI_AD(18)	11.85
AH05	DE_BA(1)	8.61
AH07	D0_Data(10)	7.7
AH09	D0_DQS(3)	6.7
AH17	D3_Data(11)	6.1
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
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
AJ12	$\overline{\text{DB\_CAS}}$	7.8
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	$\overline{\text{D2\_WE}}$	8.61
AJ25	$\overline{\text{D6\_WE}}$	8.7
AJ26	D6_Data(12)	7.9
AJ27	D6_Addr(07)	9.1
AJ28	D6_Addr(09)	10.4

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 7 of 18)

Grid Position	Signal Name	Total InCap
AJ29	PCI_AD(29)	10.65
AJ30	PCI_AD(28)	10.95
AJ31	PCI_AD(27)	11.35
AJ32	PCI_AD(26)	12.15
AJ33	PCI_AD(25)	12.25
AK03	$\overline{\text{DE\_RAS}}$	9.91
AK05	D0_Data(15)	8
AK07	D0_Data(30)	8.1
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	$\overline{\text{DE\_CAS}}$	10.61
AL03	D0_Data(12)	9.2
AL04	D0_Data(08)	8.8
AL05	$\overline{\text{IEW\_Clk\_A}}$	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
AL15	D3_Data(06)	5.1
AL17	D3_Data(04)	7.5
AL18	$\overline{\text{D3\_CS}}$	8.91
AL19	D3_Addr(09)	9.11
AL20	D2_Data(05)	7.7
AL21	D2_Addr(09)	8.91
AL22	$\overline{\text{D2\_CS}}$	9.21
AL23	D6_Data(00)	7.9

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 8 of 18)

Grid Position	Signal Name	Total InCap
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	$\overline{\text{D6\_CS}}$	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	$\overline{\text{D0\_WE}}$	9.71
AM11	D0_Addr(08)	9.51
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
AN02	D0_Data(04)	10.1
AN03	D0_Data(07)	9.9
AN04	D0_Data(17)	9.8
AN05	IEW_Clk_A	7.77
AN06	D0_Addr(03)	10.81
AN07	D0_Data(18)	9
AN08	D0_Data(24)	9.1
AN09	$\overline{\text{D0\_CS}}$	9.41
AN10	D0_Addr(01)	9.91
AN17	D3_Addr(10)	9.91
AN18	D2_Data(00)	8.5

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 9 of 18)

Grid Position	Signal Name	Total InCap
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	$\overline{\text{DA\_CAS}}$	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
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	IEW_Clk_B	7.87

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 10 of 18)

Grid Position	Signal Name	Total InCap
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	$\overline{\text{DS1\_WE}}$	8.61
C18	DS1_Data(27)	7.7
C19	DS1_DQS(1)	7.9
C20	DS1_Data(21)	7.7
C21	$\overline{\text{DC\_RAS}}$	8.7
C22	DS0_Addr(11)	9.21
C23	DS0_Addr(06)	9.11
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
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

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 11 of 18)

Grid Position	Signal Name	Total InCap
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
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	$\overline{\text{D4\_CS}}$	8.11
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	$\overline{\text{DS1\_CS}}$	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	$\overline{\text{Reset}}$	9.25
E30	DMU_D(04)	9.55
E31	DMU_D(29)	9.95
E32	DMU_D(12)	10.75

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 12 of 18)

Grid Position	Signal Name	Total InCap
F05	SCH_Addr(07)	8.61
F07	SCH_Addr(08)	8.91
F09	SCH_Data(02)	7.91
F11	$\overline{\text{D4\_WE}}$	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
F29	DMU_D(07)	8.85
F31	DMU_D(06)	10.25
F33	DMU_D(05)	11.05
G03	Spare_Tst_Rcvr(5)	7.51
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	$\overline{\text{DD\_CAS}}$	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

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 13 of 18)

Grid Position	Signal Name	Total InCap
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
H07	SCH_Addr(06)	7.91
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
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

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 14 of 18)

Grid Position	Signal Name	Total InCap
J23	$\overline{\text{DS0\_WE}}$	7.31
J24	DS0_Data(18)	6.3
J25	DMU_D(25)	7.15
J26	DMU_D(24)	7.35
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
K07	Boot_Picocode	6.98
K13	$\overline{\text{DD\_RAS}}$	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
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
L27	DMU_C(06)	7.25

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 15 of 18)

Grid Position	Signal Name	Total InCap
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
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
N06	PIO(2)	12.51
N07	cam_cp_response(13)	8.31
N08	PIO(1)	12.31
N09	PIO(0)	7.11
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	$\overline{\text{DC\_CAS}}$	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 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 16 of 18)

Grid Position	Signal Name	Total InCap
N31	DMU_C(19)	8.95
N32	DMU_C(18)	9.55
N33	DMU_C(17)	9.95
P05	cam_cp_response(2)	9.91
P07	cam_cp_response(5)	9.31
P09	cam_cp_response(9)	7.41
P15	D4_Data(28)	5.2
P19	DS0_Data(00)	5.3
P29	DMU_C(30)	8.15
P31	DMU_C(29)	8.75
P33	DMU_C(28)	9.85
R01	cam_cp_response(7)	11.51
R04	LU_Addr(08)	8.01
R05	LU_Data(33)	7.51
R06	cam_cp_response(3)	8.51
R07	LU_Data(04)	7.21
R08	LU_Data(05)	6.81
R09	cam_cp_response(10)	6.61
R10	cam_cp_response(8)	6.01
R13	cam_cp_response(12)	5.31
R14	cam_cp_response(11)	5.21
R17	D4_Data(14)	5.6
R18	DS1_DQS(3)	5.6
T01	Spare_Tst_Rcvr(3)	7.52
T03	Spare_Tst_Rcvr(8)	6.61
T05	cam_cp_response(6)	11.91
T07	LU_Data(03)	7.21
T09	LU_Data(02)	6.61
T11	cam_cp_response(1)	5.71
T13	cam_cp_response(4)	4.51
T15	LU_Data(24)	7.21
T19	Send_Grant_B	5.95
T21	RES_Data	5.55
T25	$\overline{\text{JTAG\_TRst}}$	6.85
U01	LU_Data(30)	9.41
U03	LU_Data(35)	8.11
U05	Spare_Tst_Rcvr(0)	5.47

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 17 of 18)

Grid Position	Signal Name	Total InCap
U06	Testmode(1)	5.78
U08	LU_Data(08)	6.61
U09	cam_cp_response(0)	9.91
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
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
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
V27	DMU_A(02)	6.95
V29	DMU_A(01)	7.65
V31	DMU_A(00)	8.95
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
W17	D3_Data(08)	5.7

Table 14-2. Input Capacitance (pF) (T_A = 25°C, f = 1 MHz, V_{DD} = 3.3 V ± 0.3 V) (Page 18 of 18)

Grid Position	Signal Name	Total InCap
W18	D2_Data(02)	5.6
W20	Send_Grant_A	5.65
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
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
Y05	LU_Data(32)	7.51
Y07	LU_Data(17)	7.41
Y09	LU_Data(22)	6.81
Y11	LU_Addr(01)	5.81
Y19	D2_Data(15)	5.3
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 14-3. Operating Supply Voltages

Symbol	Parameter	Rating			Units	Notes
		Min	Typ	Max		
V_{DD25} V_{DD3} V_{DD4} V_{DD5}	2.5 V Power supply	2.375	2.5	2.625	V	1
V_{DD33} V_{DD2}	3.3 V Power supply	3.135	3.3	3.465	V	1
V_{DD}	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, 2
$V_{REF1(2-0)}$ $V_{REF2(8-0)}$	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
   $V_{DD18} \leq V_{DD25} + 0.4$ V
   $V_{DD25} \leq V_{DD33} + 0.4$ V
   $V_{DD33} \leq V_{DD25} + 1.9$ V
   $V_{DD33} \leq V_{DD18} + 2.7$ V
2. See also *PLL Filter Circuit* on page 80.

Table 14-4. Operating Supply Currents

Power Supply	Min	Nominal	Max	Units
1.8 V		3.20		A
2.5 V		0.46		A
3.3 V		0.08		A

Table 14-5. Thermal Characteristics

Thermal Characteristic	Min	Nominal	Max	Units
Estimated power dissipation		7.3	8.6	W
Operating junction temperature (Tj)[1]	0		105	°C

1. Operation up to T_j = 125°C is supported for up to 3600 hours. However, the electromigration (EM) limit must not exceed 105°C for 88 KPOH equivalent. Contact your IBM field applications engineer for EM equivalents.

14.1 Driver Specifications

Table 14-6. 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 14-7. 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 14-3* (Typical = 1.8 V).

Table 14-8. 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 14-9. 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 14-3* (Typical = 2.5 V).

Table 14-10. 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 14-11. 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 14-3* (Typical = 2.5 V).

Table 14-12. 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 14-3* (Typical = 3.3 V).

Table 14-13. 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 14-3* (Typical = 3.3 V).

Table 14-14. 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 14-15. 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]	0.65 V_{DD}[3]	0.35 V_{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 14-3* (Typical = 1.8 V).

Table 14-16. 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 14-3* (Typical = 2.5 V).

Table 14-17. 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 14-3* (Typical = 3.3 V).

Table 14-18. 3.3 V LVTTL / 5 V 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

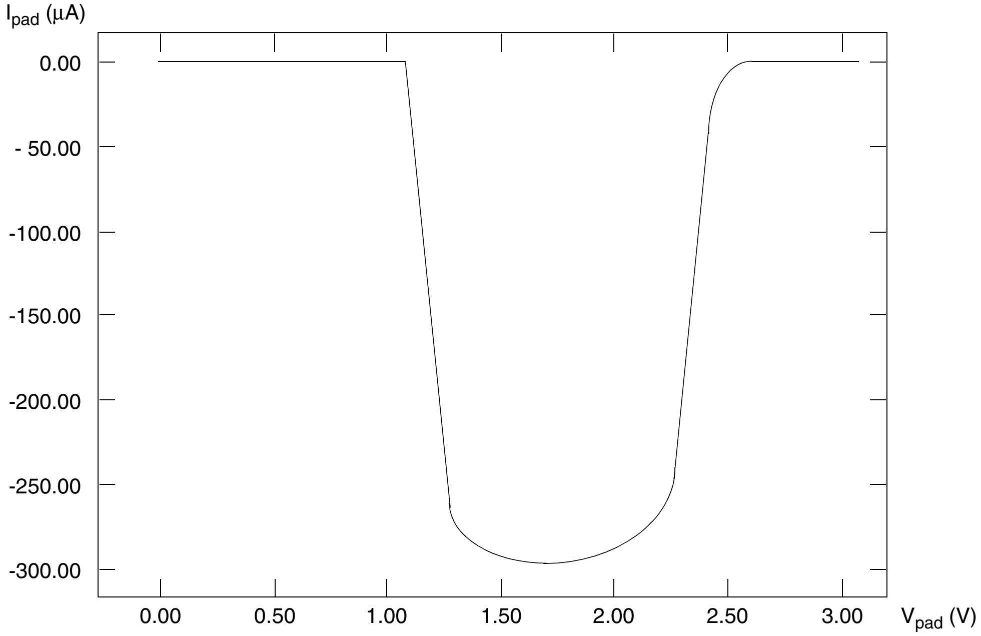
1. Maximum allowable applies to overshoot only.
2. Minimum allowable applies to undershoot only.

Table 14-19. Receiver Maximum Input Leakage DC Current Input Specifications

Function	I_{IL} (µA)	I_{IH} (µA)
Without pull-up element or pull-down element	-25 at V_{IN} = LPDL	25 at V_{IN} = MPUL
With pull-down element	-25 at V_{IN} = LPDL	200 at V_{IN} = MPUL
With pull-up element	-150 at V_{IN} = LPDL	25 at V_{IN} = MPUL

1. See section *3.3 V LVTTL / 5 V Tolerant BP33 and IP33 Receiver Input Current/Voltage Curve on page 525*.

Figure 14-1. 3.3 V LVTTL / 5 V 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 14-20. 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	
Ri	Input impedance	80	100	120	Ω	

Notes:

1. All DC characteristics are based on power supply and temperature ranges as specified above.
2. This receiver using a V_{DD} of 1.8 V nominal is compatible with the 1.5 V specification described in the LVDS standard: 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 14-21. SSTL2 DC Specifications (Page 1 of 2)

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 @ 1.95 V
V_{OL} (Class II)	Output low voltage			0.55	V	I_{OL} = 15.2 mA @ 0.55 V
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 14-21. SSTL2 DC Specifications (Page 2 of 2)

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
CBA	control block address
CDM	Cell Data Mover
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

Term	Definition
CRC	See *cyclic redundancy check.*
CS	Control Store
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
dashboard	A section of the scalar register space in the EPC where all threads can access a copy of a master value.
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

Term	Definition
EDS	Enqueuer / Dequeuer / Scheduler
E-DS	Egress Data Store
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
even parity	Data checking method where a parity bit is added to a data field. Even parity is achieved when the number of bits (data + parity) that contain a ‘1’ is even.
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 $0 < W \pm 1$. 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*.

Term	Definition
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).
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 Media-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 device 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*.

Term	Definition
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
I-CDM	Ingress Cell Data Mover
ICMP	Internet Control Message Protocol
I-DS	Ingress Data Store
IEW	Ingress-to-Egress Wrap; wraps frames from the ingress side to the egress side.
IMS	Interface Manager Services
Ingress EDS	Ingress Enqueuer / Dequeuer / Scheduler
Ingress GDQ	Ingress General Data Queue
IPC	interprocess communication
I-PMM	Ingress-Physical MAC Multiplexer
IPPS	Internet Protocol Version 4 Protocol Services
IPv4	Internet Protocol Version 4
K	2^{10}, or 1024 in decimal notation
Kb	kilobit
KB	kilobyte
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 port 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

Term	Definition
LSb	least significant bit
LSB	least significant byte
LSP	label-switched path
LU_Def	look-up definition
M	2^{20}, or 1 048 576 in decimal notation
MAC	medium access control
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.*
NBT	next bit to test
NFA	next frame control block (FCB) address

Term	Definition
NLA	next leaf address
NLS	normal latency sustainable bandwidth
NP	Network Processor
NPA	next PCSB address pointer
NPASM	IBM Network Processor Assembler
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	phase-locked loop
PMA	physical medium attachment
PMM	Physical MAC Multiplexer
PolCB	Policing Control Block

Term	Definition
POS	packet over SONET (also IP 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.
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.
ps	picosecond
PSCB	pattern search control block
PTL	Physical Transport Layer
PTS	Physical Transport Services
QCB	queue control block
QoS	See *quality of service.*
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.
R/W	Access type “Read/Write”
Rbuf	raw buffer
RCB	Reassembly Control Block
RCLR	read current leaf from rope
RED	random early detection
RMON	Remote Network Monitoring
rope	leaves in a tree linked together in a circular list

Term	Definition
RPC	remote procedure call
RTP	Real-Time Transport Protocol
RUM	Access type “reset under mask”
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.
SCB	Scheduler control block
SCI	Switch Cell Interface
SMII	Serial Media-Independent Interface
SMT	software-managed tree
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”
TBI	Ten-Bit Interface
TCP	Transmission Control Protocol
TDMU	target DMU
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, DataPool, and so on).

Term	**Definition**
TLIR	tree leaf insert rope
TLV	type length vectors
TMS	Table Management Services
TP	target port
TS	Transport Services
TSDQFL	Tree Search Dequeue Free List
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
April 9, 2002	Initial release of datasheet for IBM32NP160EPXCAA133 (revision 00).
February 12, 2003	Revision 01 of Datasheet for IBM32NP160EPXCAA133. Section 2. Physical Description Section 3. Physical MAC Multiplexer *Table 3-5. Receive Counter RAM Addresses for Ingress POS MAC*: Corrected signal name (Rx_err). Section 7. Embedded Processor Complex Globally corrected number of coprocessors (ten). *Table 7-66. Counter Coprocessor Address Map*: updated CtrControl description. *7.4.8 Counter Coprocessor*: Clarified definition of CtrControl input; changed “BlockIndex” input to “SetIndex.” Section 14. Electrical and Thermal Specifications *Table 14-19. Receiver Maximum Input Leakage DC Current Input Specifications*: Updated values. Section 15. Glossary Added “dashboard.”