PCIEX.book Page 105 Tuesday, August 5, 2003 4:22 PM

3 Address Spaces & Transaction Routing

The Previous Chapter

The previous chapter introduced the PCI Express data transfer protocol. It described the layered approach to PCI Express device design while describing the function of each device layer. Packet types employed in accomplishing data transfers were described without getting into packet content details. Finally, this chapter outlined the process of a requester initiating a transaction such as a memory read to read data from a completer across a Link.

This Chapter

This chapter describes the general concepts of PCI Express transaction routing and the mechanisms used by a device in deciding whether to accept, forward, or reject a packet arriving at an ingress port. Because Data Link Layer Packets (DLLPs) and Physical Layer *ordered set* link traffic are never forwarded, the emphasis here is on Transaction Layer Packet (TLP) types and the three routing methods associated with them: address routing, ID routing, and implicit routing. Included is a summary of configuration methods used in PCI Express to set up PCI-compatible plug-and-play addressing within system IO and memory maps, as well as key elements in the PCI Express packet protocol used in making routing decisions.

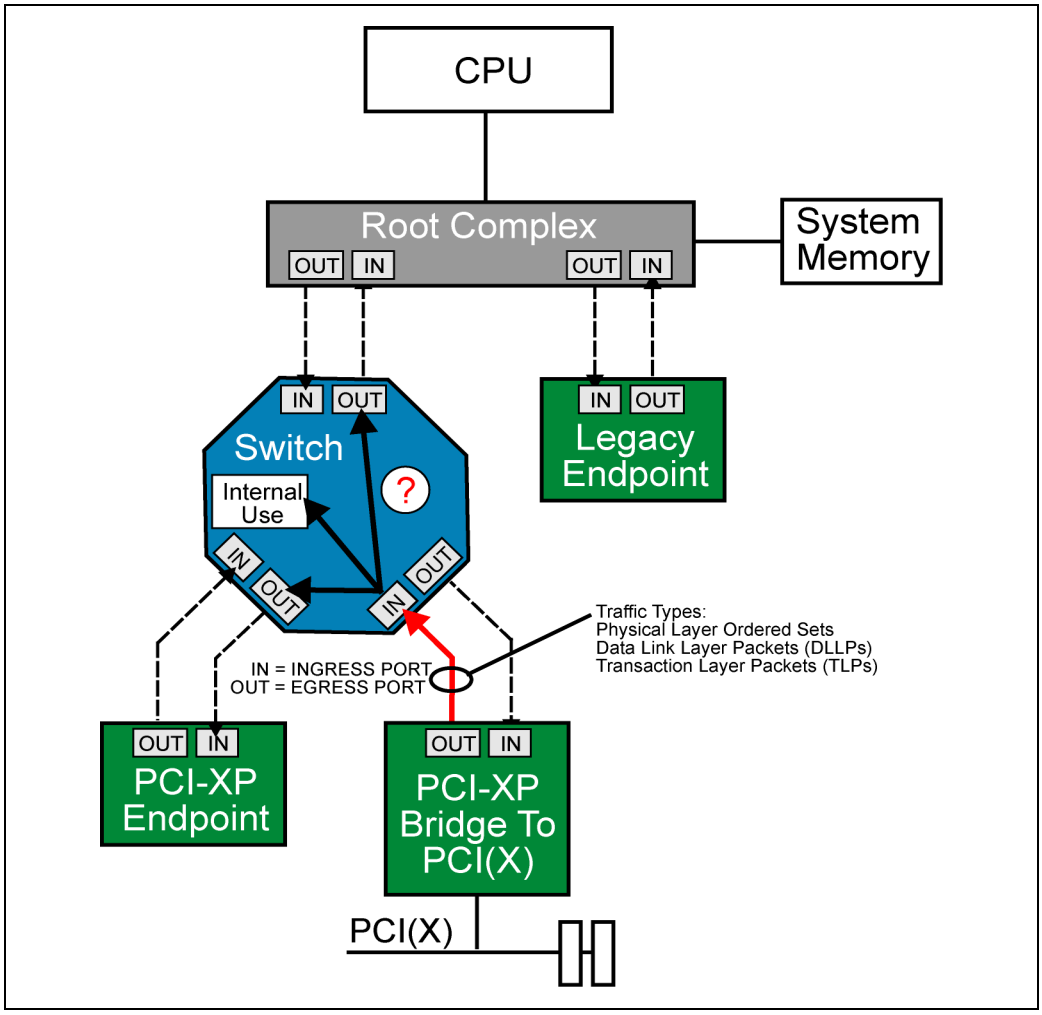
The Next Chapter

The next chapter details the two major classes of packets are *Transaction Layer Packets* (TLPs), and *Data Link Layer Packets* (DLLPs). The use and format of each TLP and DLLP packet type is covered, along with definitions of the field within the packets.

Introduction

Unlike shared-bus architectures such as PCI and PCI-X, where traffic is visible to each device and routing is mainly a concern of bridges, PCI Express devices are dependent on each other to accept traffic or forward it in the direction of the ultimate recipient.

Figure 3-1: Multi-Port PCI Express Devices Have Routing Responsibilities



Chapter 3: Address Spaces & Transaction Routing

As illustrated in Figure 3-1 on page 106, a PCI Express topology consists of independent, point-to-point links connecting each device with one or more neighbors. As traffic arrives at the inbound side of a link interface (called the *ingress port*), the device checks for errors, then makes one of three decisions:

1. Accept the traffic and use it internally.
2. Forward the traffic to the appropriate outbound *(egress*) port.
3. Reject the traffic because it is neither the intended target nor an interface to it (note that there are also other reasons why traffic may be rejected)

Receivers Check For Three Types of Link Traffic

Assuming a link is fully operational, the physical layer receiver interface of each device is prepared to monitor the logical idle condition and detect the arrival of the three types of link traffic: Ordered Sets, DLLPs, and TLPs. Using control (K) symbols which accompany the traffic to determine framing boundaries and traffic type, PCI Express devices then make a distinction between traffic which is local to the link vs. traffic which may require routing to other links (e.g. TLPs). Local link traffic, which includes Ordered Sets and Data Link Layer Packets (DLLPs), isn't forwarded and carries no routing information. Transaction Layer Packets (TLPs) can and do move from link to link, using routing information contained in the packet headers.

Multi-port Devices Assume the Routing Burden

It should be apparent in Figure 3-1 on page 106 that devices with multiple PCI Express ports are responsible for handling their own traffic as well as forwarding other traffic between ingress ports and any enabled egress ports. Also note that while peer-peer transaction support is required of switches, it is optional for a multi-port Root Complex. It is up to the system designer to account for peer-to-peer traffic when selecting devices and laying out a motherboard.

Endpoints Have Limited Routing Responsibilities

It should also be apparent in Figure 3-1 on page 106 that endpoint devices have a single link interface and lack the ability to route inbound traffic to other links. For this reason, and because they don't reside on shared busses, endpoints never expect to see ingress port traffic which is not intended for them (this is different than shared-bus PCI(X), where devices commonly decode addresses

PCIEX.book Page 108 Tuesday, August 5, 2003 4:22 PM

and commands not targeting them). Endpoint routing is limited to accepting or rejecting transactions presented to them.

System Routing Strategy Is Programmed

Before transactions can be generated by a requester, accepted by the completer, and forwarded by any devices in the path between the two, all devices must be configured to enforce the system transaction routing scheme. Routing is based on traffic type, system memory and IO address assignments, etc. In keeping with PCI plug-and-play configuration methods, each PCI express device is discovered, memory and IO address resources are assigned to them, and switch/bridge devices are programmed to forward transactions on their behalf. Once routing is programmed, bus mastering and target address decoding are enabled. Thereafter, devices are prepared to generate, accept, forward, or reject transactions as necessary.

Two Types of Local Link Traffic

Local traffic occurs between the transmit interface of one device and the receive interface of its neighbor for the purpose of managing the link itself. This traffic is never forwarded or flow controlled; when sent, it must be accepted. Local traffic is further classified as *Ordered Sets* exchanged between the Physical Layers of two devices on a link or Data Link Layer packets (DLLPs) exchanged between the Data Link Layers of the two devices.

Ordered Sets

These are sent by each physical layer transmitter to the physical layer of the corresponding receiver to initiate link training, compensate for clock tolerance, or transition a link to and from the Electrical Idle state. As indicated in Table 3-1 on page 109, there are five types of Ordered Sets.

Each ordered set is constructed of 10-bit control (K) symbols that are created within the physical layer. These symbols have a common name as well as a alph-numeric code that defines the 10 bits pattern of 1s and 0s, of which they are comprised. For example, the SKP (Skip) symbol has a 10-bit value represented as K28.0.

Figure 3-2 on page 110 illustrates the transmission of Ordered Sets. Note that each ordered set is fixed in size, consisting of 4 or 16 characters. Again, the receiver is required to consume them as they are sent. Note that the COM control symbol (K28.5) is used to indicate the start of any ordered set.

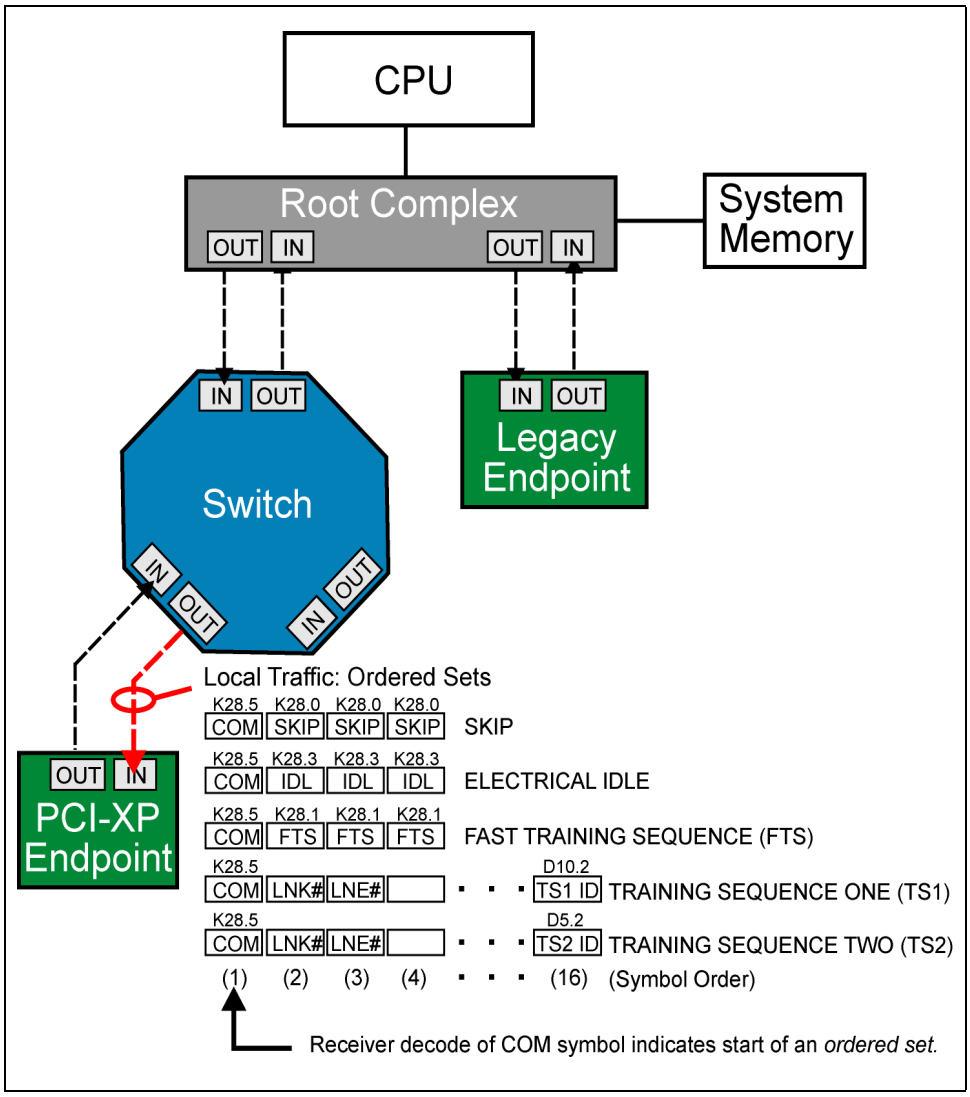
Refer to the "8b/10b Encoding" on page 419 for a thorough discussion of Ordered Sets.

Table 3-1: Ordered Set Types

Ordered Set Type	Symbols	Purpose
Fast Training Sequence (FTS)	COM, 3 FTS	Quick synchronization of bit stream when leaving L0s power state.
Training Sequence One (TS1)	COM, Lane ID, 14 more	Used in link training, to align and synchronize the incoming bit stream at startup, convey reset, other functions.
Training Sequence Two (TS2)	COM, Lane ID, 14 more	See TS1.
Electrical Idle (IDLE)	COM, 3 IDL	Indicates that link should be brought to a lower power state (L0s, L1, L2).
Skip	COM, 3 SKP	Inserted periodically to compensate for clock tolerances.

PCI Express System Architecture

Figure 3-2: PCI Express Link Local Traffic: Ordered Sets



Data Link Layer Packets (DLLPs)

The other type of local traffic sent by a device transmit interface to the corresponding receiver of the device attached to it are Data Link Layer Packets (DLLPs). These are also used in link management, although they are sourced at the device Data Link Layer rather that the Physical Layer. The main functions of DLLPs are to facilitate Link Power Management, TLP Flow Control, and the acknowledgement of successful TLP delivery across the link.

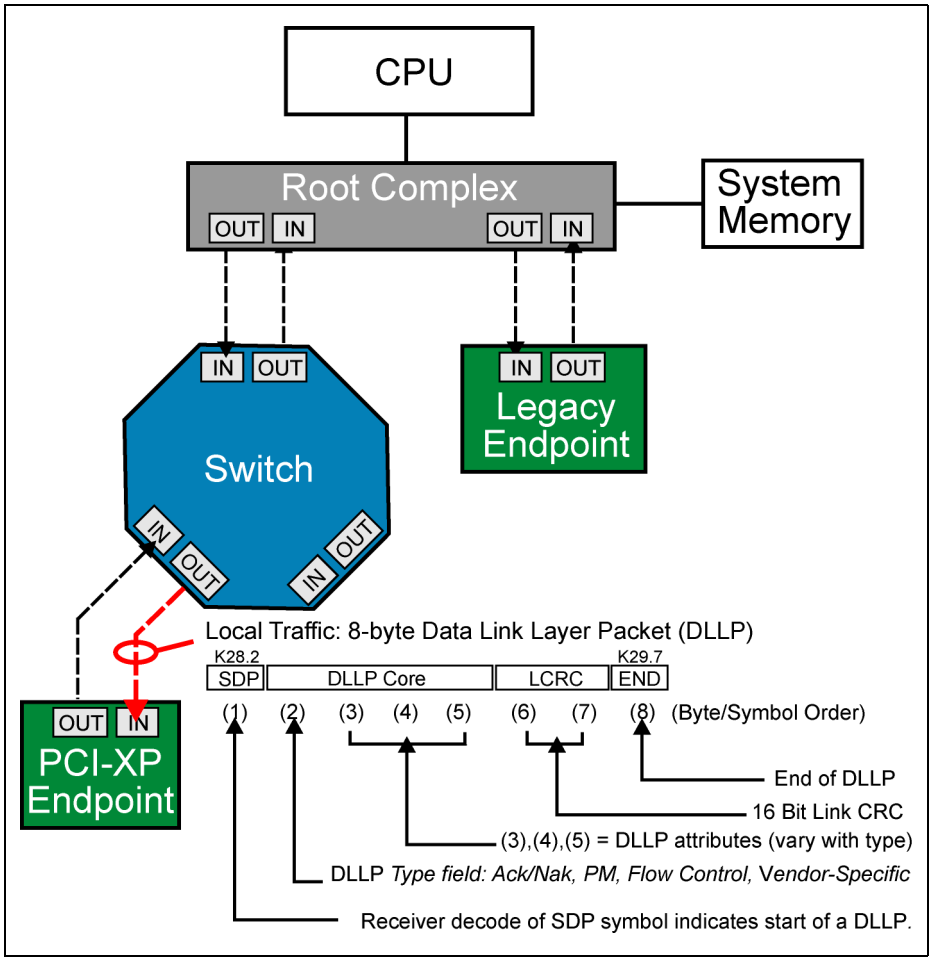
Table 3-2: Data Link Layer Packet (DLLP) Types

DLLP	Purpose
Acknowledge (Ack)	Receiver Data Link Layer sends Ack to indicate that no CRC or other errors have been encountered in received TLP(s). Transmitter retains copy of TLPs until Ack'd
No Acknowledge (Nak)	Receiver Data Link Layer sends Nak to indicate that a TLP was received with a CRC or other error. All TLPs remaining in the transmitter's Retry Buffer must be resent, in the original order.
PM_Enter_L1; PM_Enter_L23	Following a software configuration space access to cause a device power management event, a downstream device requests entry to link L1 or Level 2-3 state
PM_Active_State_Req_L1	Downstream device autonomously requests L1 Active State
PM_Request_Ack	Upstream device acknowledges transition to L1 State
Vendor-Specific DLLP	Reserved for vendor-specific purposes
InitFC1-P InitFC1-NP InitFC1-Cpl	Flow Control Initialization Type One DLLP awarding posted (P), nonposted (NP), or completion (Cpl) flow control credits.
InitFC2-P InitFC2-NP InitFC2-Cpl	Flow Control Initialization Type Two DLLP confirming award of InitFC1 posted (P), nonposted (NP), or completion (Cpl) flow control credits.
UpdateFC-P UpdateFC-NP UpdateFC-Cpl	Flow Control Credit Update DLLP awarding posted (P), nonposted (NP), or completion (Cpl) flow control credits.

PCIEX.book Page 112 Tuesday, August 5, 2003 4:22 PM

As described in Table 3-2 on page 111 and shown in Figure 3-3 on page 112, there are three major types of DLLPs: Ack/Nak, Power Management (several variants), and Flow Control. In addition, a vendor-specific DLLP is permitted in the specification. Each DLLP is 8 bytes, including a Start Of DLLP (SDP) byte, 2-byte CRC, and an End Of Packet (END) byte in addition to the 4 byte DLLP core (which includes the *type* field and any required attributes).

Figure 3-3: PCI Express Link Local Traffic: DLLPs



Note that unlike Ordered Sets, DLLPs always carry a 16-bit CRC which is verified by the receiver before carrying out the required operation. If an error is detected by the receiver of a DLLP, it is dropped. Even though DLLPs are not acknowledged, time-out mechanisms built into the specification permit recovery from dropped DLLPs due to CRC errors.

Refer to "Data Link Layer Packets" on page 198 for a thorough discussion of Data Link Layer packets.

Transaction Layer Packet Routing Basics

The third class of link traffic originates in the Transaction Layer of one device and targets the Transaction Layer of another device. These Transaction Layer Packets (TLPs) are forwarded from one link to another as necessary, subject to the routing mechanisms and rules described in the following sections. Note that other chapters in this book describe additional aspects of Transaction Layer Packet handling, including Flow Control, Quality Of Service, Error Handling, Ordering rules, etc. The term transaction is used here to describe the exchange of information using Transaction Layer Packets. Because Ordered Sets and DLLPs carry no routing information and are not forwarded, the routing rules described in the following sections apply only to TLPs.

TLPs Used to Access Four Address Spaces

As transactions are carried out between PCI Express requesters and completers, four separate address spaces are used: Memory, IO, Configuration, and Message. The basic use of each address space is described in Table 3-3 on page 113.

Table 3-3: PCI Express Address Space And Transaction Types

Address Space	Transaction Types	Purpose
Memory	Read, Write	Transfer data to or from a location in the system memory map
IO	Read, Write	Transfer data to or from a location in the system IO map
Configuration	Read, Write	Transfer data to or from a location in the configuration space of a PCI-compatible device.

Table 3-3: PCI Express Address Space And Transaction Types (Continued)

Address Space	Transaction Types	Purpose
Message	Baseline, Vendor-specific	General in-band messaging and event reporting (without consuming memory or IO address resources)

Split Transaction Protocol Is Used

Accesses to the four address spaces in PCI Express are accomplished using split-transaction requests and completions.

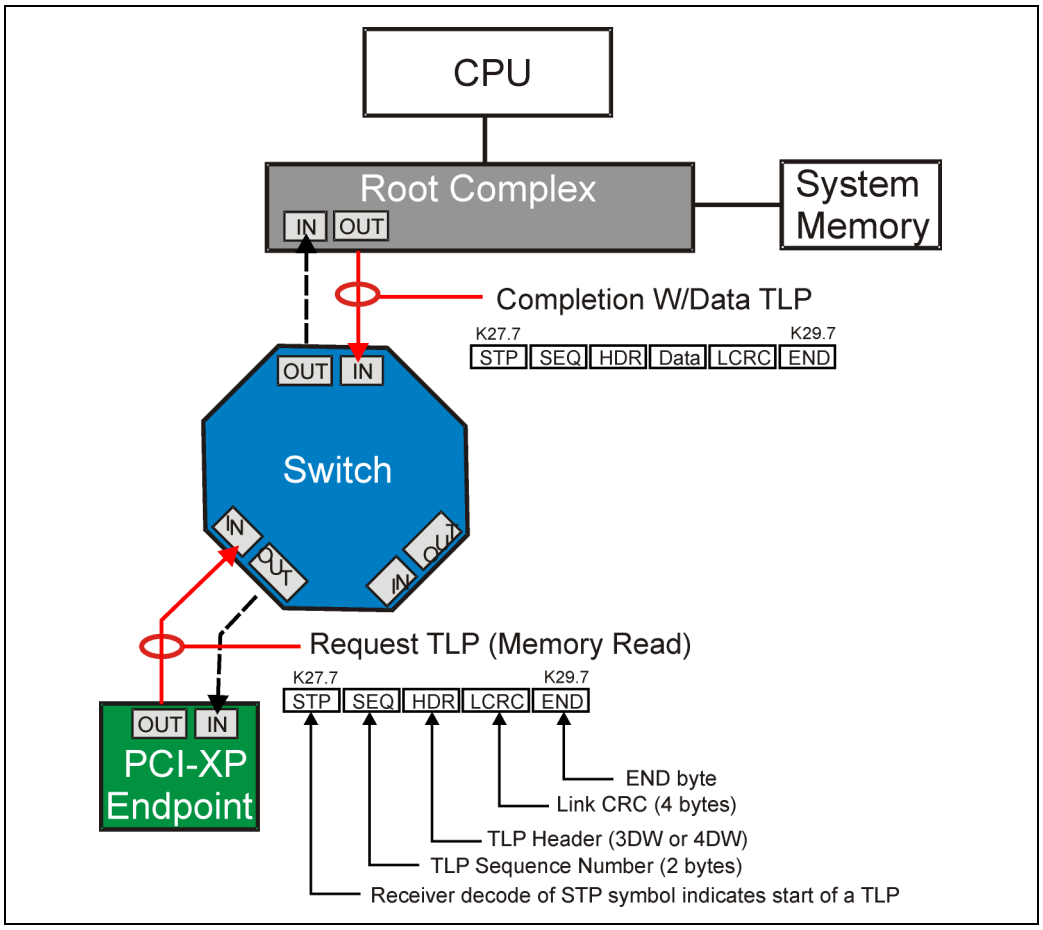
Split Transactions: Better Performance, More Overhead

The split transaction protocol is an improvement over earlier bus protocols (e.g. PCI) which made extensive use of bus wait-states or delayed transactions (retries) to deal with latencies in accessing targets. In PCI Express, the completion following a request is initiated by the completer only when it has data and/or status ready for delivery. The fact that the completion is separated in time from the request which caused it also means that two separate TLPs are generated, with independent routing for the request TLP and the Completion TLP. Note that while a link is free for other activity in the time between a request and its subsequent completion, a split-transaction protocol involves some additional overhead as two complete TLPs must be generated to carry out a single transaction.

Figure 3-4 on page 115 illustrates the request-completion phases of a PCI Express split transaction. This example represents an endpoint read from system memory.

Chapter 3: Address Spaces & Transaction Routing

Figure 3-4: PCI Express Transaction Request And Completion TLPs



Write Posting: Sometimes a Completion Isn't Needed

To mitigate the penalty of the request-completion latency, messages and some write transactions in PCI Express are posted, meaning the write request (including data) is sent, and the transaction is over from the requester's perspective as soon as the request is sent out of the egress port; responsibility for delivery is now the problem of the next device. In a multi-level topology, this has the advantage of being much faster than waiting for the entire request-completion transit, but — as in all posting schemes — uncertainty exists concerning when (and if) the transaction completed successfully at the ultimate recipient.

In PCI Express, write posting to memory is considered acceptable in exchange for the higher performance. On the other hand, writes to IO and configuration space may change device behavior, and write posting is not permitted. A completion will always be sent to report status of the IO or configuration write operation.

Table 3-4 on page 116 lists PCI Express posted and non-posted transactions.

Table 3-4: PCI Express Posted and Non-Posted Transactions

Request	How Request Is Handled
Memory Write	All Memory Write requests are posted. No completion is expected or sent.
Memory Read Memory Read Lock	All memory read requests are non-posted. A completion with data (CplD or CplDLK) will be returned by the completer with requested data and to report status of the memory read
IO Write	All IO Write requests are non-posted. A completion without data (Cpl) will be returned by the completer to report status of the IO write operation.
IO Read	All IO read requests are non-posted. A completion with data (CplD) will be returned by the completer with requested data and to report status of the IO read operation.
Configuration Write Type 0 and Type 1	All Configuration Write requests are non-posted. A completion without data (Cpl) will be returned by the completer to report status of the configuration space write operation.
Configuration Read Type 0 and Type 1	All configuration read requests are non-posted. A completion with data (CplD) will be returned by the completer with requested data and to report status of the read operation.
Message Message With Data	While the routing method varies, all message transactions are handled in the same manner as memory writes in that they are considered posted requests

Chapter 3: Address Spaces & Transaction Routing

Three Methods of TLP Routing

All of the TLP variants, targeting any of the four address spaces, are routed using one of the three possible schemes: Address Routing, ID Routing, and Implicit Routing. Table 3-5 on page 117 summarizes the PCI Express TLP header type variants and the routing method used for each. Each of these is described in the following sections.

Table 3-5: PCI Express TLP Variants And Routing Options

TLP Type	Routing Method Used
Memory Read (MRd), Memory Read Lock (MRdLk), Memory Write (MWr)	Address Routing
IO Read (IORd), IO Write (IOWr)	Address Routing
Configuration Read Type 0 (CfgRd0), Configuration Read Type 1 (CfgRd1) Configuration Write Type 0 (CfgWr0), Configuration Write Type 1(CfgWr1)	ID Routing
Message (Msg), Message With Data (MsgD)	Address Routing, ID Routing, or Implicit routing
Completion (Cpl), Completion With Data (CplD)	ID Routing

PCI Express Routing Is Compatible with PCI

As indicated in Table 3-5 on page 117, memory and IO transactions are routed through the PCI Express topology using address routing to reference system memory and IO maps, while configuration cycles use ID routing to reference the completer's (target's) logical position within the PCI-compatible bus topology (using Bus Number, Device Number, Function Number in place of a linear address). Both address routing and ID routing are completely compatible with routing methods used in the PCI and PCIX protocols when performing memory, IO, or configuration transactions. PCI Express completions also use the ID routing scheme.

PCI Express Adds Implicit Routing for Messages

PCI Express adds the third routing method, implicit routing, which is an option when sending messages. In implicit routing, neither address or ID routing information applies; the packet is routed based on a code in the packet header indicating it is destined for device(s) with known, fixed locations (the Root Complex, the next receiver, etc.).

While limited in the cases it can support, implicit routing simplifies routing of messages. Note that messages may optionally use address or ID routing instead.

Why Were Messages Added to PCI Express Protocol? PCI and PCI-X protocols support *load and store* memory and IO read-write transactions, which have the following features:

1. The transaction initiator drives out a memory or IO start address selecting a location within the desired target.
2. The target claims the transaction based on decoding and comparing the transaction start address with ranges it has been programmed to respond to in its configuration space Base Address Registers.
3. If the transaction involves bursting, then addresses are indexed after each data transfer.

While PCI Express also supports load and store transactions with its memory and IO transactions, it adds in-band messages. The main reason for this is that the PCI Express protocol seeks to (and does) eliminate many of the sideband signals related to interrupts, error handling, and power management which are found in PCI(X)-based systems. Elimination of signals is very important in an architecture with the scalability possible with PCI Express. It would not be efficient to design a PCI Express device with a two lane link and then saddle it with numerous additional signals to handle auxiliary functions.

The PCI Express protocol replaces most sideband signals with a variety of in-band packet types; some of these are conveyed as Data Link Layer packets (DLLPs) and some as Transaction Layer packets (TLPs).

How Implicit Routing Helps with Messages. One side effect of using in-band messages in place of hard-wired sideband signals is the problem of delivering the message to the proper recipient in a topology consisting of numerous point-to-point links. The PCI Express protocol provides maximum flexibility in routing message TLPs; they may use address routing, ID routing, or the third method, implicit routing. Implicit routing takes advantage of the fact that, due to their architecture, switches and other multi-port

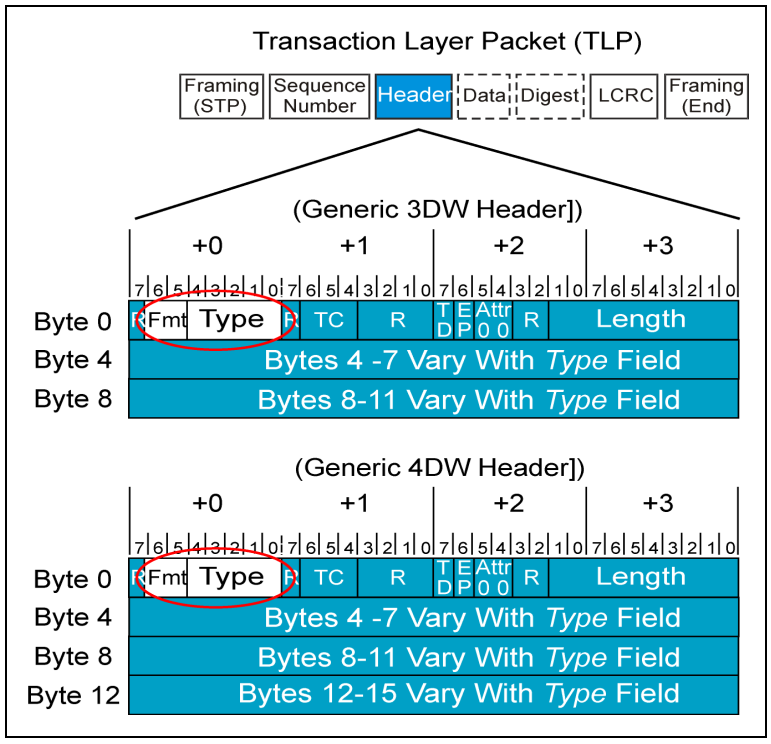
devices have a fundamental sense of upstream and downstream, and where the Root Complex is to be found. Because of this, a message header can be routed implicitly with a simple code indicating that it is intended for the Root Complex, a broadcast downstream message, should terminate at the next receiver, etc.

The advantage of implicit routing is that it eliminates the need to assign a set of memory mapped addresses for all of the possible message variants and program all of the devices to use them.

Header Fields Define Packet Format and Routing

As depicted in Figure 3-5 on page 119, each Transaction Layer Packet contains a three or four double word (12 or 16 byte) header. Included in the 3DW or 4DW header are two fields, *Type* and *Format* (Fmt), which define the format of the remainder of the header and the routing method to be used on the entire TLP as it moves between devices in the PCI Express topology.

Figure 3-5: Transaction Layer Packet Generic 3DW And 4DW Headers



PCIEX.book Page 120 Tuesday, August 5, 2003 4:22 PM

Using TLP Header Information: Overview

General

As TLPs arrive at an ingress port, they are first checked for errors at both the physical and data link layers of the receiver. Assuming there are no errors, TLP routing is performed; basic steps include:

1. The TLP header *Type* and *Format* fields in the first DWord are examined to determine the size and format of the remainder of the packet.
2. Depending on the routing method associated with the packet, the device will determine if it is the intended recipient; if so, it will accept (consume) the TLP. If it is not the recipient, and it is a multi-port device, it will forward the TLP to the appropriate egress port--subject to the rules for ordering and flow control for that egress port.
3. If it is neither the intended recipient nor a device in the path to it, it will generally reject the packet as an Unsupported Request (UR).

Header Type/Format Field Encodings

Table 3-6 on page 120 below summarizes the encodings used in TLP header Type and Format fields. These two fields, used together, indicate TLP format and routing to the receiver.

Table 3-6: TLP Header Type and Format Field Encodings

TLP	FMT[1:0]	TYPE [4:0]
Memory Read Request (MRd)	00 = 3DW, no data 01 = 4DW, no data	0 0000
Memory Read Lock Request (MRdLk)	00 = 3DW, no data 01 = 4DW, no data	0 0001
Memory Write Request (MWr)	10 = 3DW, w/ data 11 = 4DW, w/ data	0 0000
IO Read Request (IORd)	00 = 3DW, no data	00010
IO Write Request (IOWr)	10 = 3DW, w/ data	0 0010

Table 3-6: TLP Header Type and Format Field Encodings (Continued)

TLP	FMT[1:0]	TYPE [4:0]
Config Type 0 Read Request (CfgRd0)	00 = 3DW, no data	0 0100
Config Type 0 Write Request (CfgWr0)	10 = 3DW, w/ data	0 0100
Config Type 1 Read Request (CfgRd1)	00 = 3DW, no data	0 0101
Config Type 1 Write Request (CfgWr1)	10 = 3DW, w/ data	0 0101
Message Request (Msg)	01 = 4DW, no data	1 0 RRR* (for RRR, see routing subfield)
Message Request W/Data (MsgD)	11 = 4DW, w/ data	1 0 RRR* (for RRR, see routing subfield)
Completion (Cpl)	00 = 3DW, no data	0 1010
Completion W/Data (CplD)	10 = 3DW, w/ data	0 1010
Completion-Locked (CplLk)	00 = 3DW, no data	0 1011
Completion W/Data (CplDLk)	10 = 3DW, w/ data	0 1011

Applying Routing Mechanisms

Once configuration of the system routing strategy is complete and transactions are enabled, PCI Express devices decode inbound TLP headers and use corresponding fields in configuration space Base Address Registers, Base/Limit registers, and Bus Number registers to apply address, ID, and implicit routing to the packet. Note that there are actually two levels of decision: the device first determines if the packet targets an internal location; if not, and the device is a switch, it will evaluate the packet to see if it should be forwarded out of an egress port. A third possibility is that the packet has been received in error or is malformed; in this case, it will be handled as a receive error. There are a number of cases when this may happen, and a number of ways it may be handled. Refer to "PCI Express Error Checking Mechanisms" on page 356 for a description of error checking and handling. The following sections describe the basic features of each routing mechanism; we will assume no errors are encountered.

PCIEX.book Page 122 Tuesday, August 5, 2003 4:22 PM

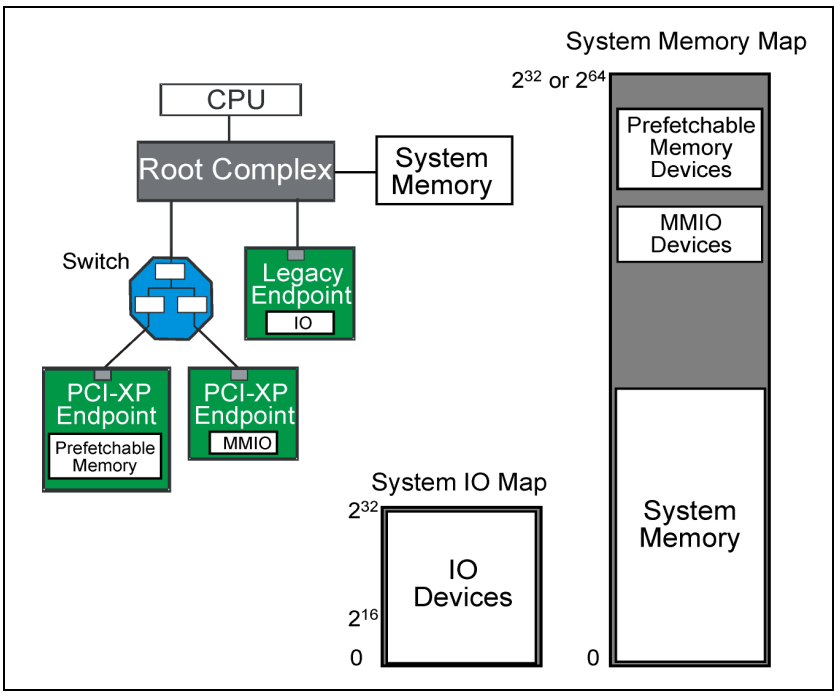
Address Routing

PCI Express transactions using address routing reference the same system memory and IO maps that PCI and PCIX transactions do. Address routing is used to transfer data to or from memory, memory mapped IO, or IO locations. Memory transaction requests may carry either 32 bit addresses using the 3DW TLP header format, or 64 bit addresses using the 4DW TLP header format. IO transaction requests are restricted to 32 bits of address using the 3DW TLP header format, and should only target legacy devices.

Memory and IO Address Maps

Figure 3-6 on page 122 depicts generic system memory and IO maps. Note that the size of the system memory map is a function of the range of addresses that devices are capable of generating (often dictated by the CPU address bus). As in PCI and PCI-X, PCI Express permits either 32 bit or 64 bit memory addressing. The size of the system IO map is limited to 32 bits (4GB), although in many systems only the lower 16 bits (64KB) are used.

Figure 3-6: Generic System Memory And IO Address Maps



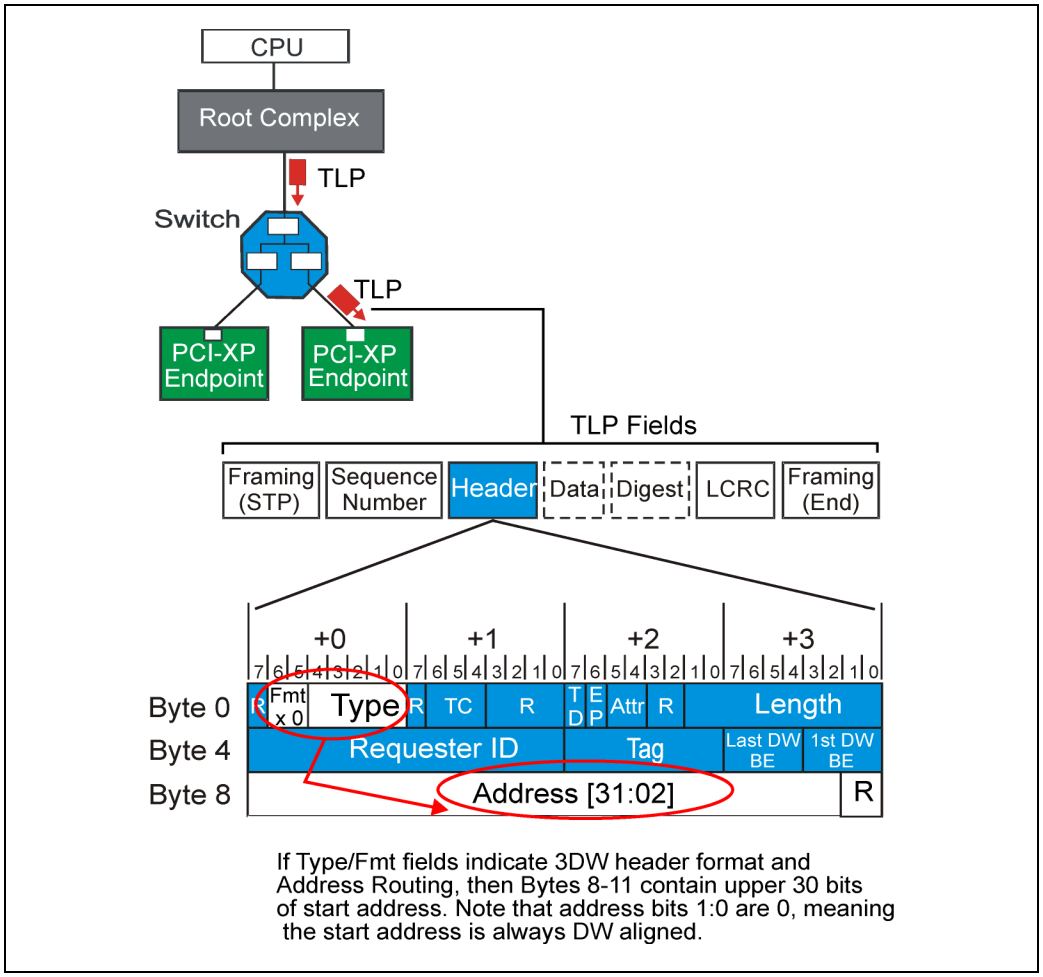
Chapter 3: Address Spaces & Transaction Routing

Key TLP Header Fields in Address Routing

If the Type field in a received TLP indicates address routing is to be used, then the Address Fields in the header are used to performing the routing check. There are two cases: 32-bit addresses and 64-bit addresses.

TLPs with 3DW, 32-Bit Address. For IO or a 32-bit memory requests, only 32 bits of address are contained in the header. Devices targeted with these TLPs will reside below the 4GB memory or IO address boundary. Figure 3-7 on page 123 depicts this case.

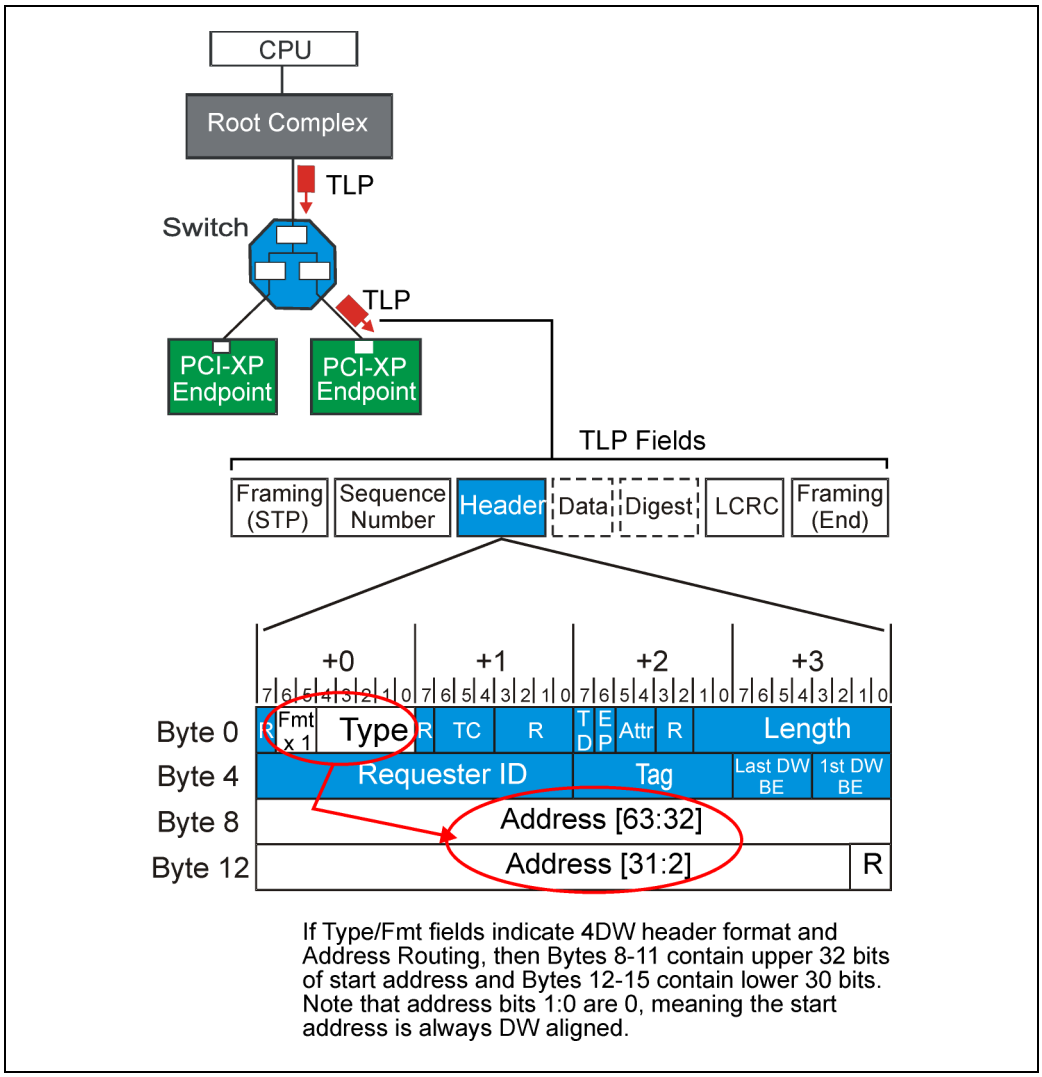
Figure 3-7: 3DW TLP Header Address Routing Fields



PCIEX.book Page 124 Tuesday, August 5, 2003 4:22 PM

TLPs With 4DW, 64-Bit Address. For 64-bit memory requests, 64 bits of address are contained in the header. Devices targeted with these TLPs will reside above the 4GB memory boundary. Figure 3-8 on page 124 shows this case.

Figure 3-8: 4DW TLP Header Address Routing Fields

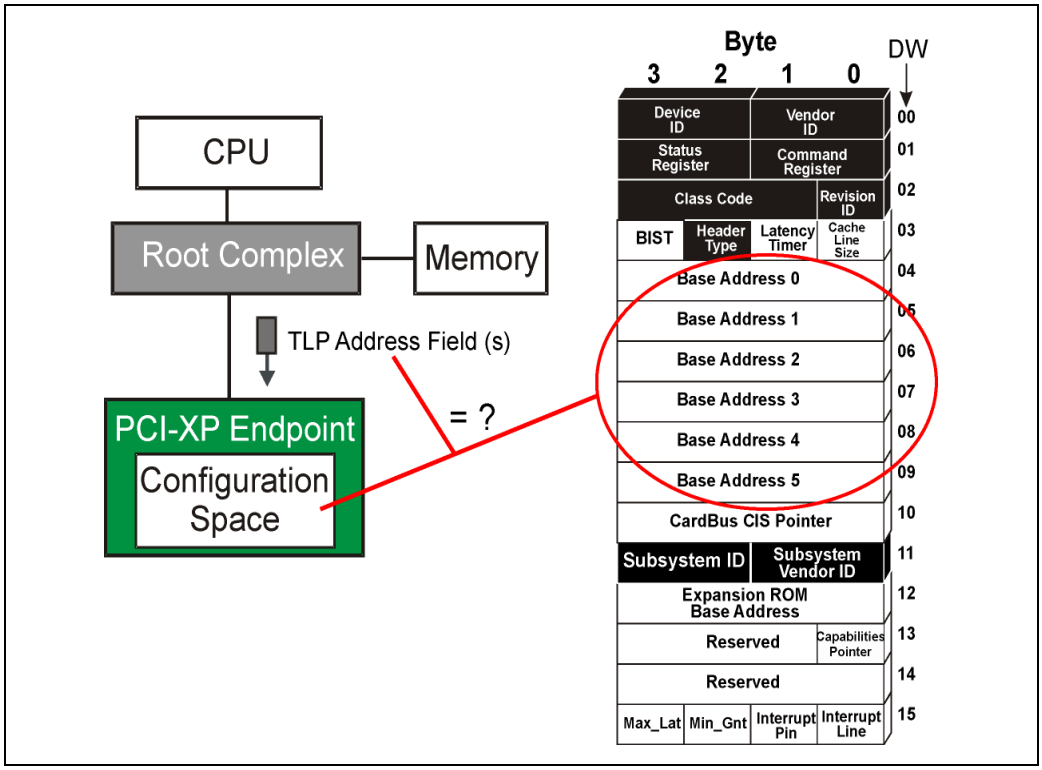


Chapter 3: Address Spaces & Transaction Routing

An Endpoint Checks an Address-Routed TLP

If the Type field in a received TLP indicates address routing is to be used, then an endpoint device simply checks the address in the packet header against each of its implemented BARs in its Type 0 configuration space header. As it has only one link interface, it will either claim the packet or reject it. Figure 3-9 on page 125 illustrates this case.

Figure 3-9: Endpoint Checks Routing Of An Inbound TLP Using Address Routing



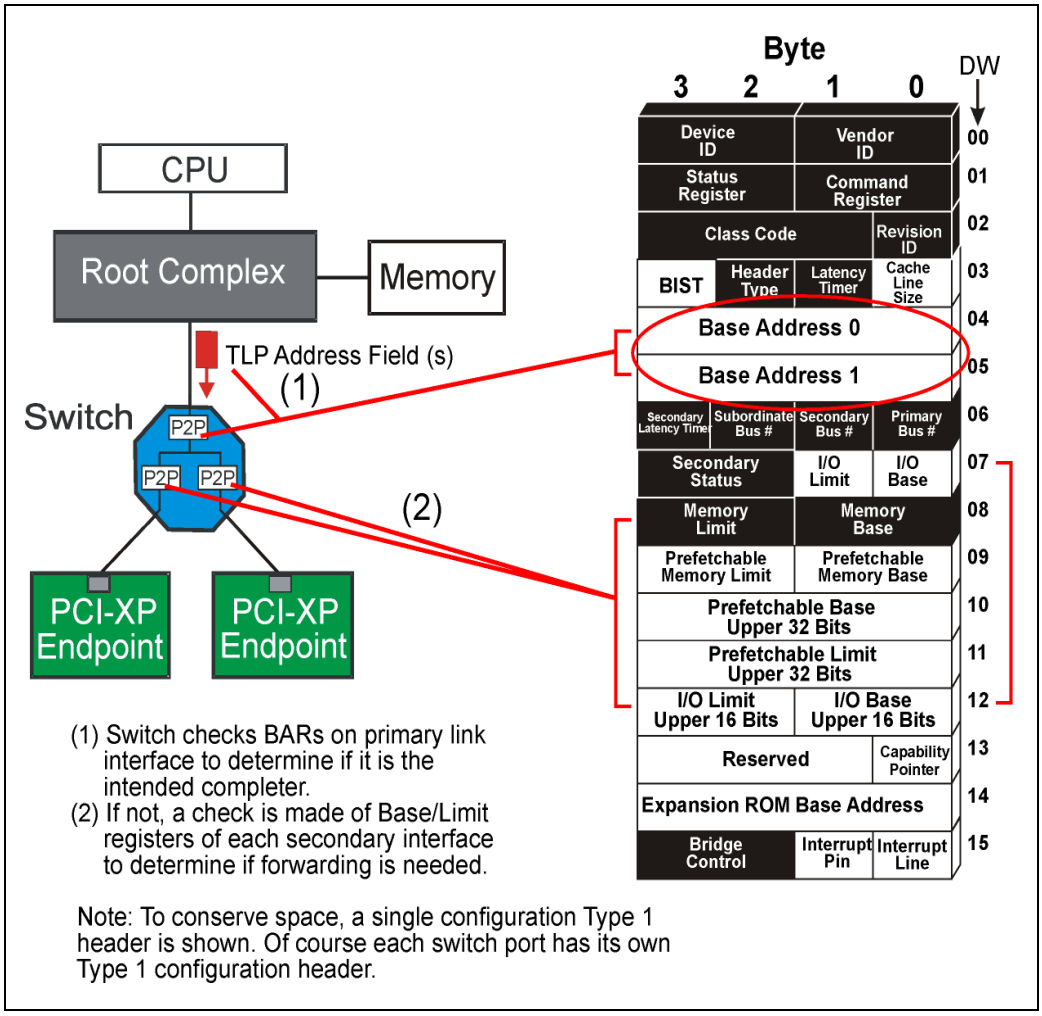
A Switch Receives an Address Routed TLP: Two Checks

General. If the Type field in a received TLP indicates address routing is to be used, then a switch first checks to see if it is the intended completer. It compares the header address against target addresses programmed in its two BARs. If the address falls within the range, it consumes the packet. This

PCIEX.book Page 126 Tuesday, August 5, 2003 4:22 PM

case is indicated by (1) in Figure 3-10 on page 126. If the header address field does not match a range programmed in a BAR, it then checks the Type 1 configuration space header for each downstream link. It checks the non-prefetchable memory (MMIO) and prefetchable Base/Limit registers if the transaction targets memory, or the I/O Base and Limt registers if the transaction targets I/O address space. This check is indicated by (2) in Figure 3-10 on page 126.

Figure 3-10: Switch Checks Routing Of An Inbound TLP Using Address Routing



Other Notes About Switch Address-Routing. The following notes also apply to switch address routing:

1. If the address-routed packet address falls in the range of one of its secondary bridge interface Base/Limit register sets, it will forward the packet downstream.
2. If the address-routed packet was moving downstream (was received on the primary interface) and it does not map to any BAR or downstream link Base/Limit registers, it will be handled as an unsupported request on the primary link.
3. Upstream address-routed packets are always forwarded to the upstream link if they do not target an internal location or another downstream link.

ID Routing

ID routing is based on the logical position (Bus Number, Device Number, Function Number) of a device function within the PCI bus topology. ID routing is compatible with routing methods used in the PCI and PCIX protocols when performing Type 0 or Type 1 configuration transactions. In PCI Express, it is also used for routing completions and may be used in message routing as well.

ID Bus Number, Device Number, Function Number Limits

PCI Express supports the same basic topology limits as PCI and PCI-X:

1. A maximum of 256 busses/links in a system. This includes busses created by bridges to other PCI-compatible protocols such as PCI, PCI-X, AGP, etc.
2. A maximum of 32 devices per bus/link. Of course, While a PCI(X) bus or the internal bus of a switch may host more than one downstream bridge interface, external PCI Express links are always point-to-point with only two devices per link. The downstream device on an external link is device 0.

- A maximum of 8 internal functions per device.

A significant difference in PCI Express over PCI is the provision for extending the amount of configuration space per function from 256 bytes to 4KB. Refer to the "Configuration Overview" on page 711 for a detailed description of the compatible and extended areas of PCI Express configuration space.

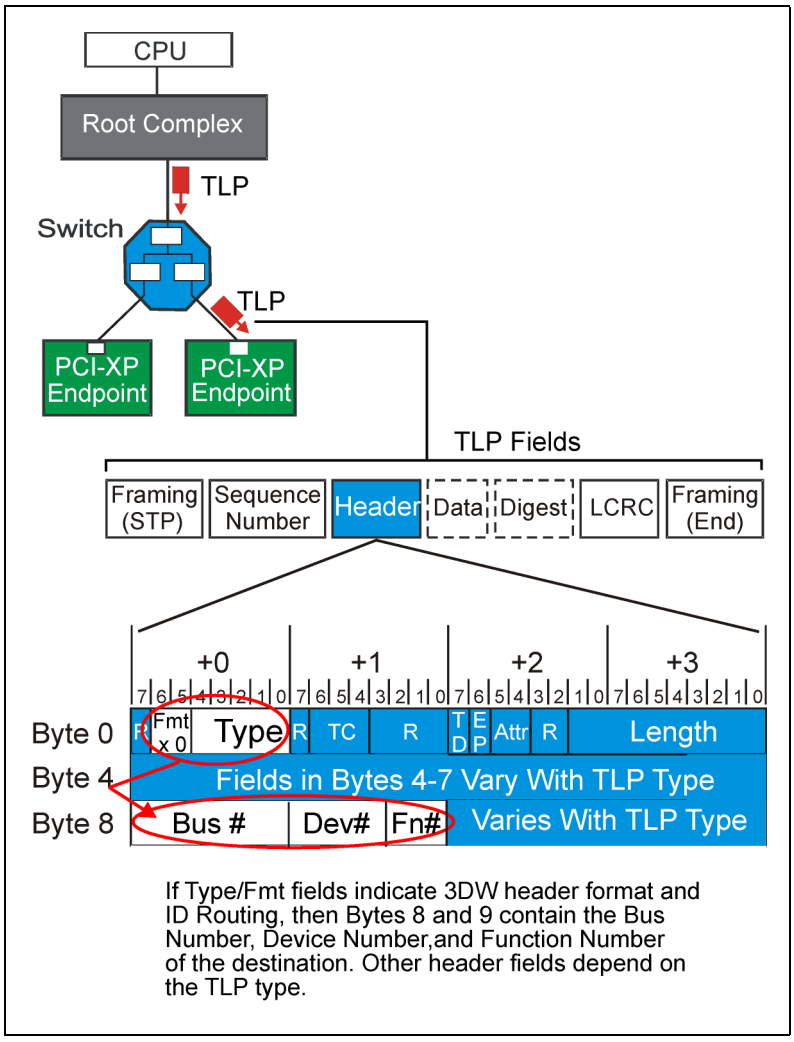
PCIEX.book Page 128 Tuesday, August 5, 2003 4:22 PM

Key TLP Header Fields in ID Routing

If the Type field in a received TLP indicates ID routing is to be used, then the ID fields in the header are used to perform the routing check. There are two cases: ID routing with a 3DW header and ID routing with a 4DW header.

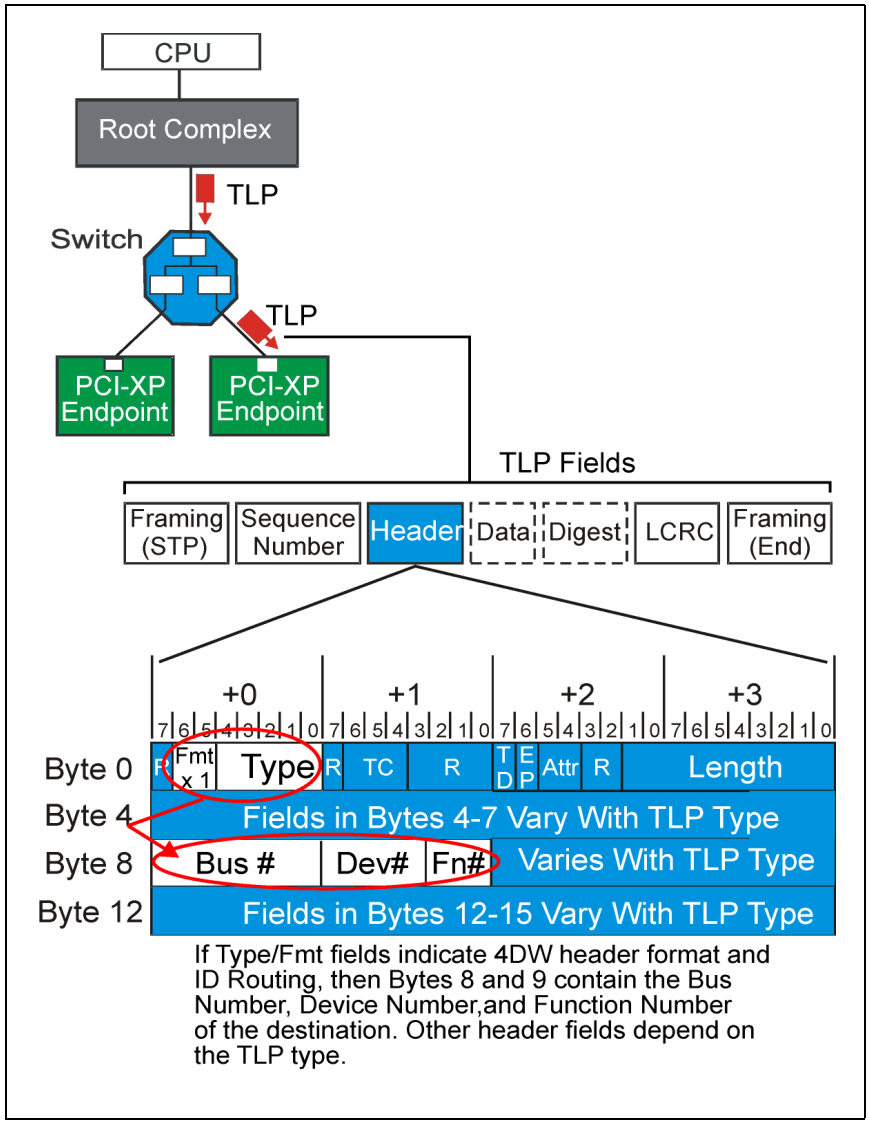
3DW TLP, ID Routing. Figure 3-11 on page 128 illustrates a TLP using ID routing and the 3DW header.

Figure 3-11: 3DW TLP Header ID Routing Fields



4DW TLP, ID Routing. Figure 3-12 on page 129 illustrates a TLP using ID routing and the 4DW header.

Figure 3-12: 4DW TLP Header ID Routing Fields



PCIEX.book Page 130 Tuesday, August 5, 2003 4:22 PM

An Endpoint Checks an ID-Routed TLP

If the Type field in a received TLP indicates ID routing is to be used, then an endpoint device simply checks the ID field in the packet header against its own Bus Number, Device Number, and Function Number(s). In PCI Express, each device "captures" (and remembers) its own Bus Number and Device Number contained in TLP header bytes 8-9 each time a configuration write (Type 0) is detected on its primary link. At reset, all bus and device numbers in the system revert to 0, so a device will not respond to transactions other than configuration cycles until at least one configuration write cycle (Type 0) has been performed. Note that the PCI Express protocol does not define a configuration space location where the device function is required to store the captured Bus Number and Device Number information, only that it must do it.

Once again, as it has only one link interface, an endpoint will either claim an ID-routed packet or reject it. Figure 3-11 on page 128 illustrates this case.

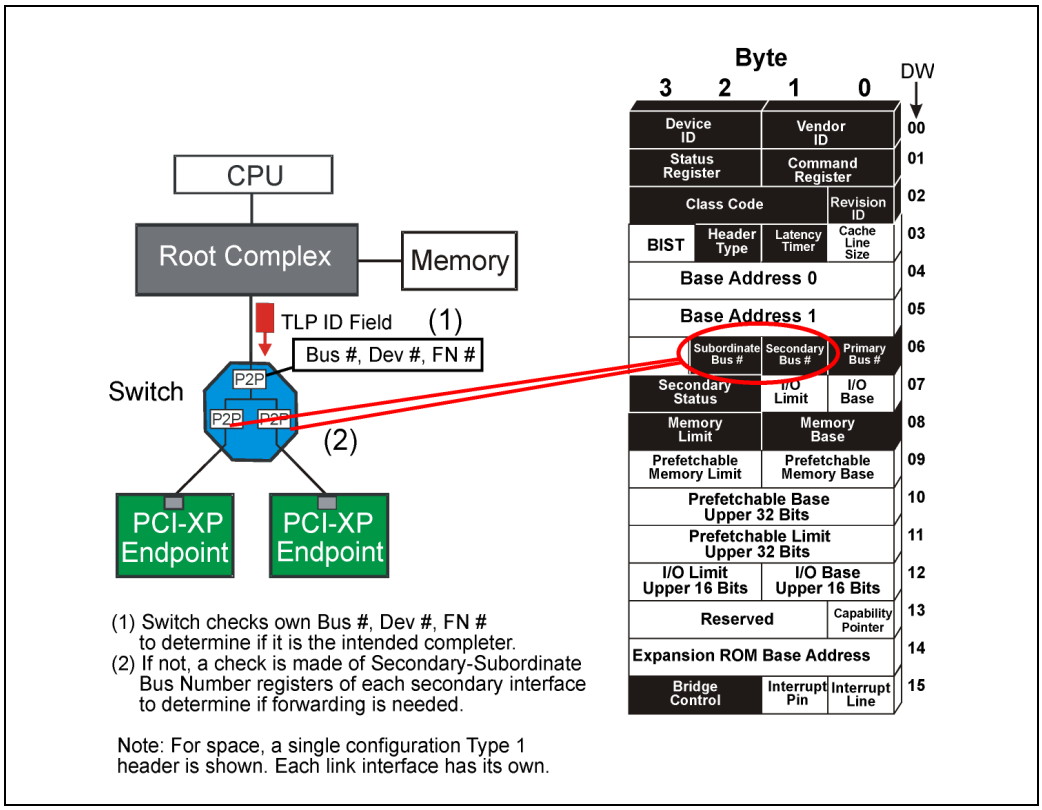
A Switch Receives an ID-Routed TLP: Two Checks

If the Type field in a received TLP indicates ID routing is to be used, then a switch first checks to see if it is the intended completer. It compares the header ID field against its own Bus Number, Device Number, and Function Number(s). This is indicated by (1) in Figure 3-13 on page 131. As in the case of an endpoint, a switch captures its own Bus Number and Device number each time a configuration write (Type 0) is detected on i's primary link interface. If the header ID agrees with the ID of the switch, it consumes the packet. If the ID field does not match i's own, it then checks the Secondary-Subordinate Bus Number registers in the configuration space for each downstream link. This check is indicated by (2) in Figure 3-13 on page 131.

Other Notes About Switch ID Routing

1. If the ID-routed packet matches the range of one of its secondary bridge interface Secondary-Subordinate registers, it will forward the packet downstream.
2. If the ID-routed packet was moving downstream (was received on the primary interface) and it does not map to any downstream interface, it will be handled as an unsupported request on the primary link.
3. Upstream ID-routed packets are always forwarded to the upstream link if they do not target an internal location or another downstream link.

Figure 3-13: Switch Checks Routing Of An Inbound TLP Using ID Routing



Implicit Routing

Implicit routing is based on the intrinsic knowledge PCI Express devices are required to have concerning upstream and downstream traffic and the existence of a single PCI Express Root Complex at the top of the PCI Express topology. This awareness allows limited routing of packets without the need to assign and include addresses with certain message packets. Because the Root Complex generally implements power management and interrupt controllers, as well as system error handling, it is either the source or recipient of most PCI Express messages.

PCIEX.book Page 132 Tuesday, August 5, 2003 4:22 PM

Only Messages May Use Implicit Routing

With the elimination of many sideband signals in the PCI Express protocol, alternate methods are required to inform the host system when devices need service with respect to interrupts, errors, power management, etc. PCI Express addresses this by defining a number of special TLPs which may be used as virtual wires in conveying sideband events. Message groups currently defined include:

- Power Management
- INTx legacy interrupt signaling
- Error signaling
- Locked Transaction support
- Hot Plug signaling
- Vendor-specific messages
- Slot Power Limit messages

Messages May Also Use Address or ID Routing

In systems where all or some of this event traffic should target the system memory map or a logical location in the PCI bus topology, address routing and ID routing may be used in place of implicit routing. If address or ID routing is chosen for a message, then the routing mechanisms just described are applied in the same way as they would for other posted write packets.

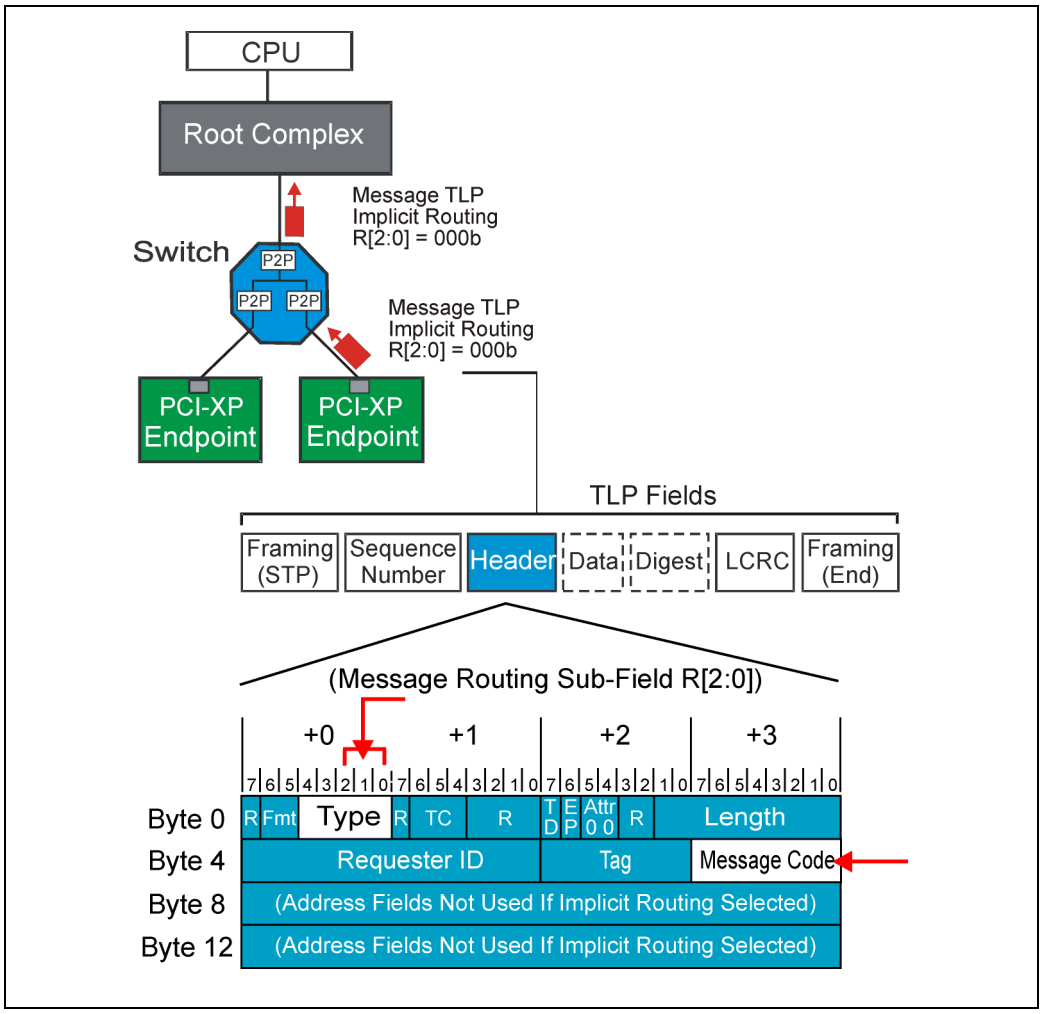
Routing Sub-Field in Header Indicates Routing Method

As a message TLP moves between PCI Express devices, packet header fields indicate both that it is a message, and whether it should be routed using address, ID, or implicitly.

Key TLP Header Fields in Implicit Routing

If the Type field in a received message TLP indicates implicit routing is to be used, then the routing sub-field in the header is also used to determine the message destination when the routing check is performed. Figure 3-14 on page 133 illustrates a message TLP using implicit routing.

Figure 3-14: 4DW Message TLP Header Implicit Routing Fields



Message Type Field Summary

Table 3-7 on page 134 summarizes the use of the TLP header Type field when a message is being sent. As shown, the upper two bits of the 5 bit Type field indicate the packet is a message, and the lower three bits are the routing sub-field which specify the routing method to apply. Note that the 4DW header is always used with message TLPs, regardless of the routing option selected.

Table 3-7: Message Request Header Type Field Usage

Type Field Bits	Description
Bit 4:3	Defines the type of transaction: 10b = Message Transaction
Bit 2:0	Message Routing Subfield R[2:0], used to select message routing: • 000b = Route to Root Complex • 001b = Use Address Routing • 010b = Use ID Routing • 011b = Route as a Broadcast Message from Root Complex • 100b = Local message; terminate at receiver (INTx messages) • 101b = Gather & route to Root Complex (PME_TO_Ack message)

An Endpoint Checks a TLP Routed Implicitly

If the Type field in a received message TLP indicates implicit routing is to be used, then an endpoint device simply checks that the routing sub-field is appropriate for it. For example, an endpoint may accept a broadcast message or a message which terminates at the receiver; it won't accept messages which implicitly target the Root Complex.

A Switch Receives a TLP Routed Implicitly

If the Type field in a received message TLP indicates implicit routing is to be used, then a switch device simply considers the ingress port it arrived on and whether the routing sub-field code is appropriate for it. Some examples:

1. The upstream link interface of a switch may legitimately receive a broadcast message routed implicitly from the Root Complex. If it does, it will forward it intact onto all downstream links. It should not see an implicitly routed broadcast message arrive on a downstream ingress port, and will handle this as a malformed TLP.
2. The switch may accept messages indicating implicit routing to the root complex on secondary links; it will forward all of these upstream because it "knows" the location of the Root Complex is on its primary side. It would not accept messages routed implicitly to the Root Complex if they arrived on the primary link receive interface.

PCIEX.book Page 135 Tuesday, August 5, 2003 4:22 PM

3. If the implicitly-routed message arrives on either upstream or downstream ingress ports, the switch may consume the packet if routing indicates it should terminate at receiver.
4. If messages are routed using address or ID methods, a switch will simply perform normal address checks in deciding whether to accept or forward it.

Plug-And-Play Configuration of Routing Options

PCI-compatible configuration space and PCI Express extended configuration space are covered in detail in the Part 6. For reference, the programming of three sets of configuration space registers related to routing is summarized here.

Routing Configuration Is PCI-Compatible

PCI Express supports the basic 256 byte PCI configuration space common to all compatible devices, including the Type 0 and Type 1 PCI configuration space header formats used by non-bridge and switch/bridge devices, respectively. Devices may implement basic PCI-equivalent functionality with no change to drivers or Operating System software.

Two Configuration Space Header Formats: Type 0, Type 1

PCI Express endpoint devices support a single PCI Express link and use the Type 0 (non-bridge) format header. Switch/bridge devices support multiple links, and implement a Type 1 format header for each link interface. Figure 3-15 on page 136 illustrates a PCI Express topology and the use of configuration space Type 0 and Type 1 header formats.

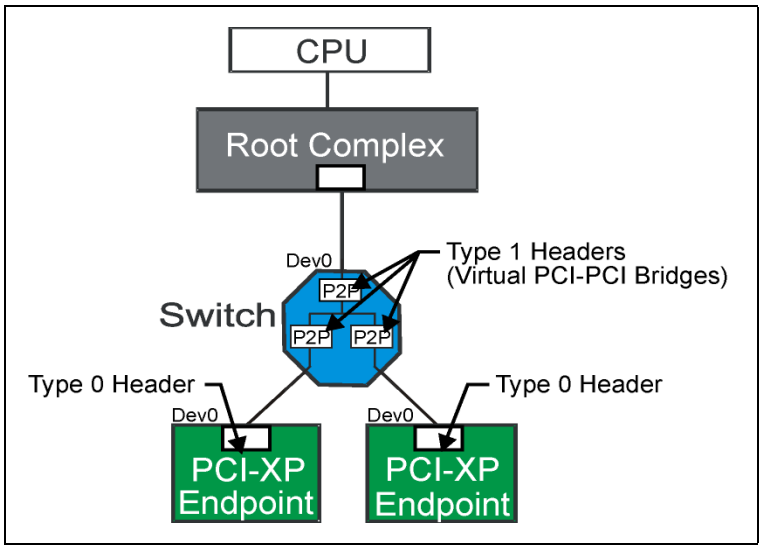
Routing Registers Are Located in Configuration Header

As with PCI, registers associated with transaction routing are located in the first 64 bytes (16 DW) of configuration space (referred to in PCI Express as the PCI 2.3 compatible header area). The three sets of registers of principal interest are:

1. *Base Address Registers (BARs)* found in Type 0 and Type 1 headers.
2. Three sets of *Base/Limit Register* pairs supported in the Type 1 header of switch/bridge devices.
3. Three *Bus Number Registers*, also found in Type 1 headers of bridge/devices.

Figure 3-16 on page 137 illustrates the Type 0 and Type 1 PCI Express Configuration Space header formats. Key routing registers are indicated.

Figure 3-15: PCI Express Devices And Type 0 And Type 1 Header Use



Base Address Registers (BARs): Type 0, 1 Headers

General

The first of the configuration space registers related to routing are the Base Address Registers (BARs) These are marked "<1" in Figure 3-16 on page 137, and are implemented by all devices which require system memory, IO, or memory mapped IO (MMIO) addresses allocated to them as targets. The location and use of BARs is compatible with PCI and PCI-X. As shown in Figure 3-16 on page 137, a Type 0 configuration space header has 6 BARs available for the device designer (at DW 4-9), while a Type 1 header has only two BARs (at DW 4-5).

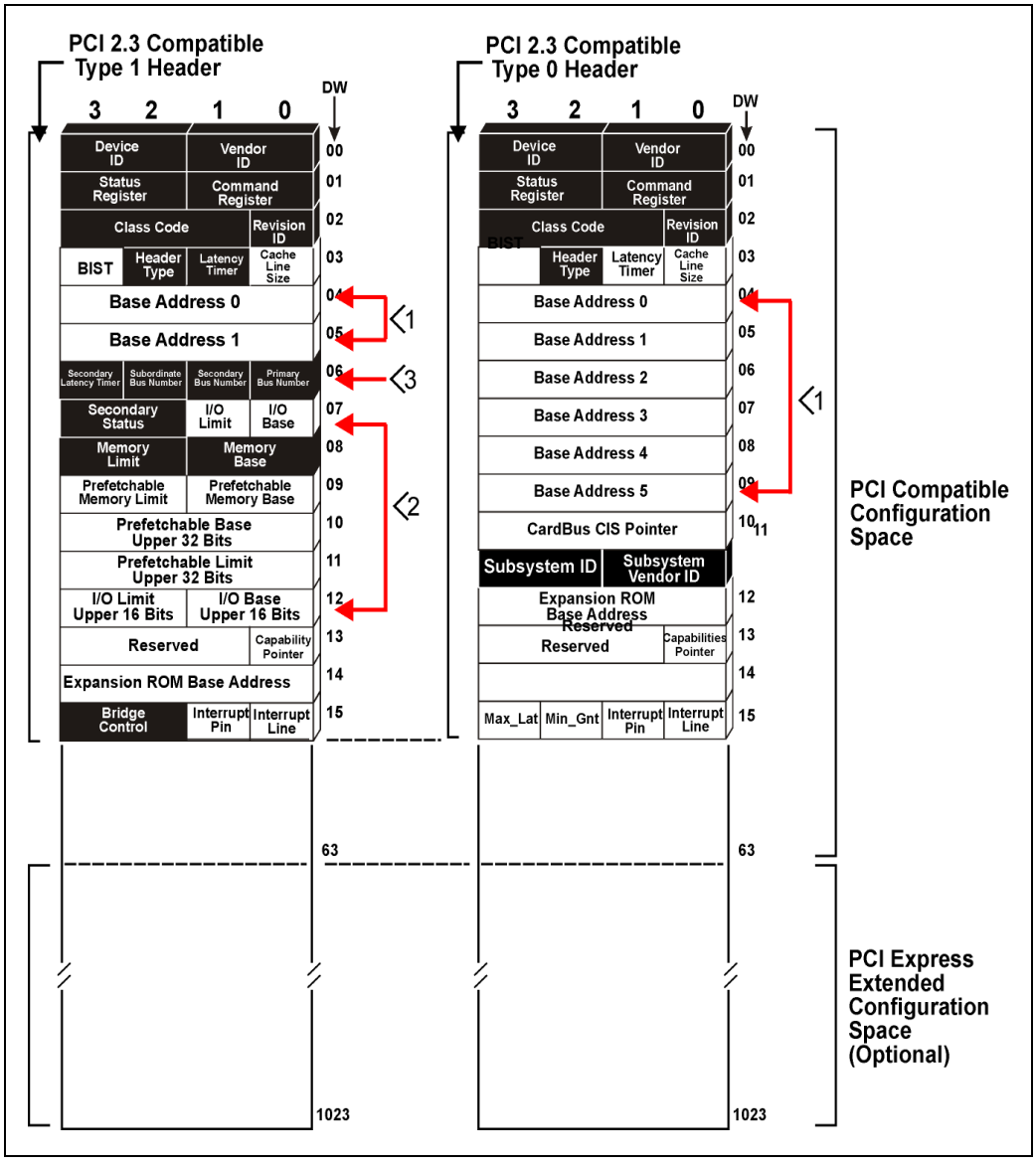
After discovering device resource requirements, system software programs each BAR with start address for a range of addresses the device may respond to as a completer (target). Set up of BARs involves several things:

1. The device designer uses a BAR to hard-code a request for an allocation of one block of prefetchable or non-prefetchable memory, or of IO addresses in the system memory or IO map. A pair of adjacent BARs are concatenated if a 64-bit memory request is being made.

PCIEX.book Page 137 Tuesday, August 5, 2003 4:22 PM

2. Hard-coded bits in the BAR include an indication of the request type, the size of the request, and whether the target device may be considered prefetchable (memory requests only).

Figure 3-16: PCI Express Configuration Space Type 0 and Type 1 Headers



During enumeration, all PCI-compatible devices are discovered and the BARs are examined by system software to decode the request. Once the system memory and IO maps are established, software programs upper bits in implemented BARs with the start address for the block allocated to the target.

BAR Setup Example One: 1MB, Prefetchable Memory Request

Figure 3-17 depicts the basic steps in setting up a BAR which is being used to track a 1 MB block of prefetchable addresses for a device residing in the system memory map. In the diagram, the BAR is shown at three points in the configuration process:

1. The uninitialized BAR in Figure 3-17 is as it looks after power-up or a reset. While the designer has tied lower bits to indicate the request type and size, there is no requirement about how the upper bits (which are read-write) must come up in a BAR, so these bits are indicated with XXXXX. System software will first write all 1's to the BAR to set all read-write bits = 1. Of course, the hard-coded lower bits are not affected by the configuration write.
2. The second view of the BAR shown in Figure 3-17 is as it looks after configuration software has performed the write of all 1's to it. The next step in configuration is a read of the BAR to check the request. Table 3-8 on page 140 summarizes the results of this configuration read.

3. The third view of the BAR shown in Figure 3-17 on page 139 is as it looks after configuration software has performed another configuration write (Type 0) to program the start address for the block. In this example, the device start address is 2GB, so bit 31 is written = 1 (2^{31} = 2GB) and all other upper bits are written = 0's.

At this point the configuration of the BAR is complete. Once software enables memory address decoding in the PCI command register, the device will claim memory transactions in the range 2GB to 2GB+1MB.

Chapter 3: Address Spaces & Transaction Routing

Figure 3-17: 32-Bit Prefetchable Memory BAR Set Up

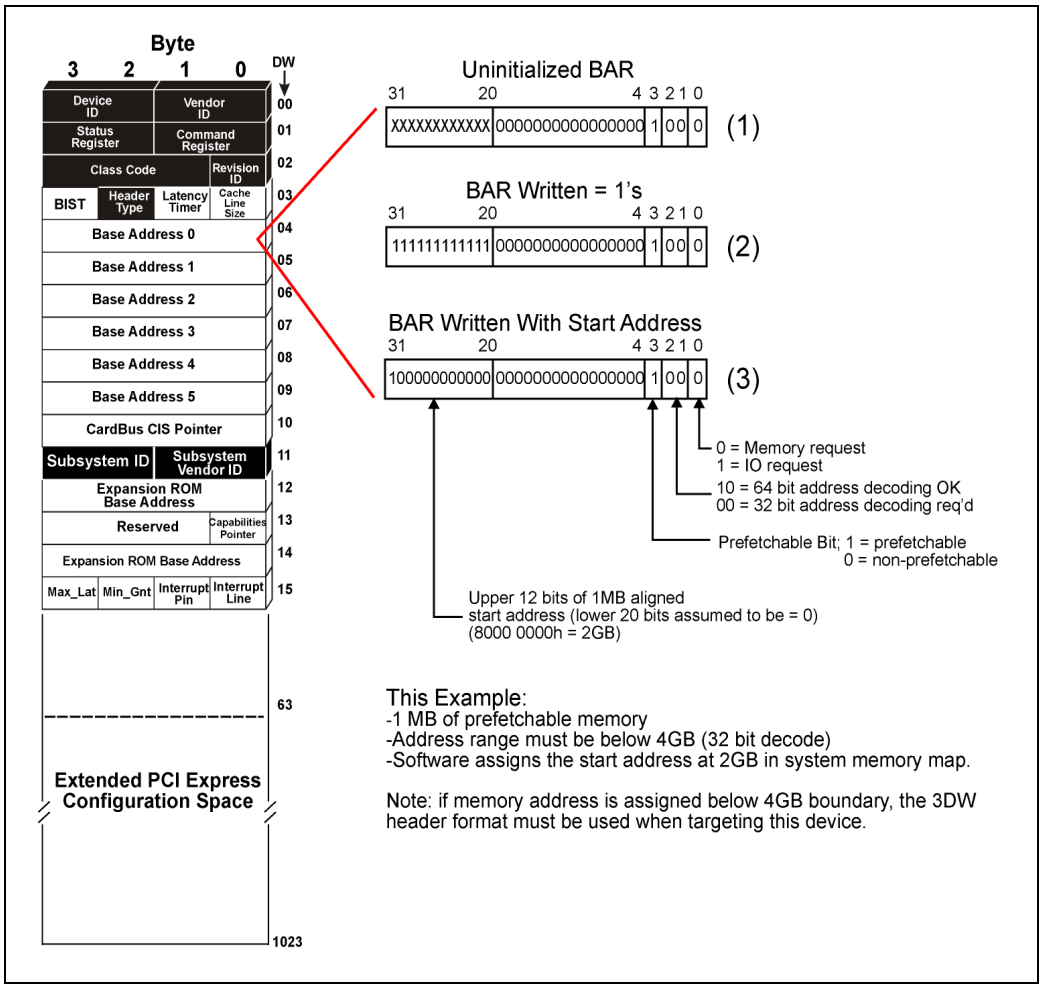


Table 3-8: Results Of Reading The BAR after Writing All "1s" To It

BAR Bits	Meaning
0	Read back as a "0", indicating a memory request
2:1	Read back as 00b indicating the target only supports a 32 bit address decoder
3	Read back as a "1", indicating request is for prefetchable memory
19:4	All read back as "0", used to help indicate the size of the request (also see bit 20)
31:20	All read back as "1" because software has not yet programmed the upper bits with a start address for the block. Note that because bit 20 was the first bit (above bit 3) to read back as written (=1); this indicates the memory request size is 1MB (2^{20} = 1MB).

BAR Setup Example Two: 64-Bit, 64MB Memory Request

Figure 3-18 on page 141 depicts the basic steps in setting up a pair of BARs being used to track a 64 MB block of prefetchable addresses for a device residing in the system memory map. In the diagram, the BARs are shown at three points in the configuration process:

1. The uninitialized BARs are as they look after power-up or a reset. The designer has hard-coded lower bits of the lower BAR to indicate the request type and size; the upper BAR bits are all read-write. System software will first write all 1's to both BARs to set all read-write bits = 1. Of course, the hard-coded bits in the lower BAR are unaffected by the configuration write.
2. The second view of the BARs in Figure 3-18 on page 141 shows them as they look after configuration software has performed the write of all 1's to both. The next step in configuration is a read of the BARs to check the request. Table 3-9 on page 142 summarizes the results of this configuration read.
3. The third view of the BAR pair Figure 3-18 on page 141 indicates conditions after configuration software has performed two configuration writes (Type 0) to program the two halves of the 64 bit start address for the block. In this example, the device start address is 16GB, so bit 1 of the Upper BAR (address bit 33 in the BAR pair) is written = 1 (2^{33} = 16GB); all other read-write bits in both BARs are written = 0's.

Chapter 3: Address Spaces & Transaction Routing

At this point the configuration of the BAR pair is complete. Once software enables memory address decoding in the PCI command register, the device will claim memory transactions in the range 16GB to 16GB+64MB.

Figure 3-18: 64-Bit Prefetchable Memory BAR Set Up

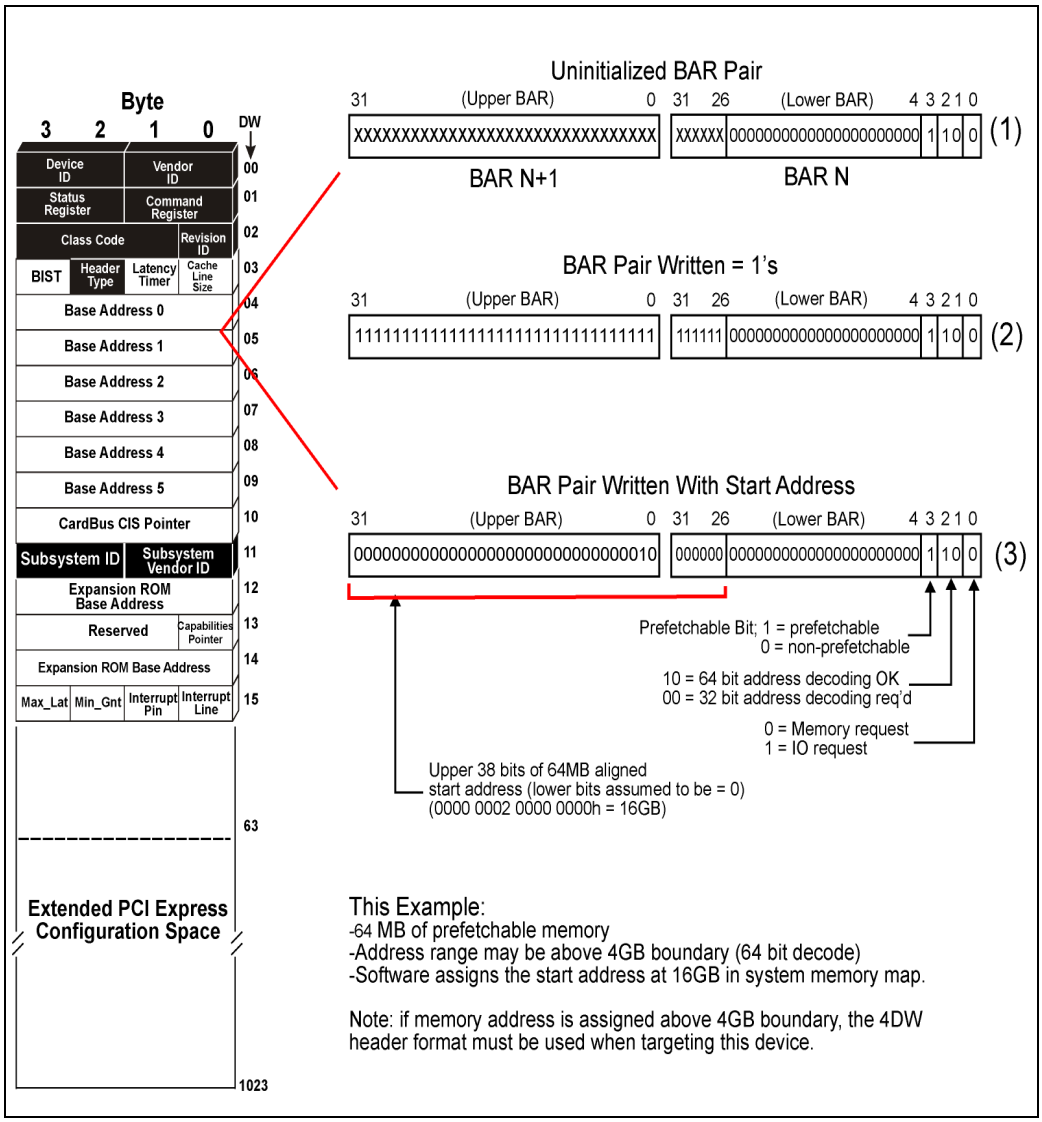


Table 3-9: Results Of Reading The BAR Pair after Writing All "1s" To Both

BAR	BAR Bits	Meaning
Lower	0	Read back as a "0", indicating a memory request
Lower	2:1	Read back as 10 b indicating the target supports a 64 bit address decoder, and that the first BAR is concatenated with the next
Lower	3	Read back as a "1", indicating request is for prefetchable memory
Lower	25:4	All read back as "0", used to help indicate the size of the request (also see bit 26)
Lower	31:26	All read back as "1" because software has not yet programmed the upper bits with a start address for the block. Note that because bit 26 was the first bit (above bit 3) to read back as written (=1); this indicates the memory request size is 64MB (2^{26} = 64MB).
Upper	31:0	All read back as "1". These bits will be used as the upper 32 bits of the 64-bit start address programmed by system software.

BAR Setup Example Three: 256-Byte IO Request

Figure 3-19 on page 143 depicts the basic steps in setting up a BAR which is being used to track a 256 byte block of IO addresses for a legacy PCI Express device residing in the system IO map. In the diagram, the BAR is shown at three points in the configuration process:

1. The uninitialized BAR in Figure 3-19 is as it looks after power-up or a reset. System software first writes all 1's to the BAR to set all read-write bits = 1. Of course, the hard-coded bits are unaffected by the configuration write.
2. The second view of the BAR shown in Figure 3-19 on page 143 is as it looks after configuration software has performed the write of all 1's to it. The next step in configuration is a read of the BAR to check the request. Table 3-10 on page 144 summarizes the results of this configuration read.
3. The third view of the BAR shown Figure 3-19 on page 143 is as it looks after configuration software has performed another configuration write (Type 0) to program the start address for the IO block. In this example, the device start address is 16KB, so bit 14 is written = 1 (2^{14} = 16KB); all other upper bits are written = 0's.

Chapter 3: Address Spaces & Transaction Routing

At this point the configuration of the IO BAR is complete. Once software enables IO address decoding in the PCI command register, the device will claim IO transactions in the range 16KB to 16KB+256.

Figure 3-19: IO BAR Set Up

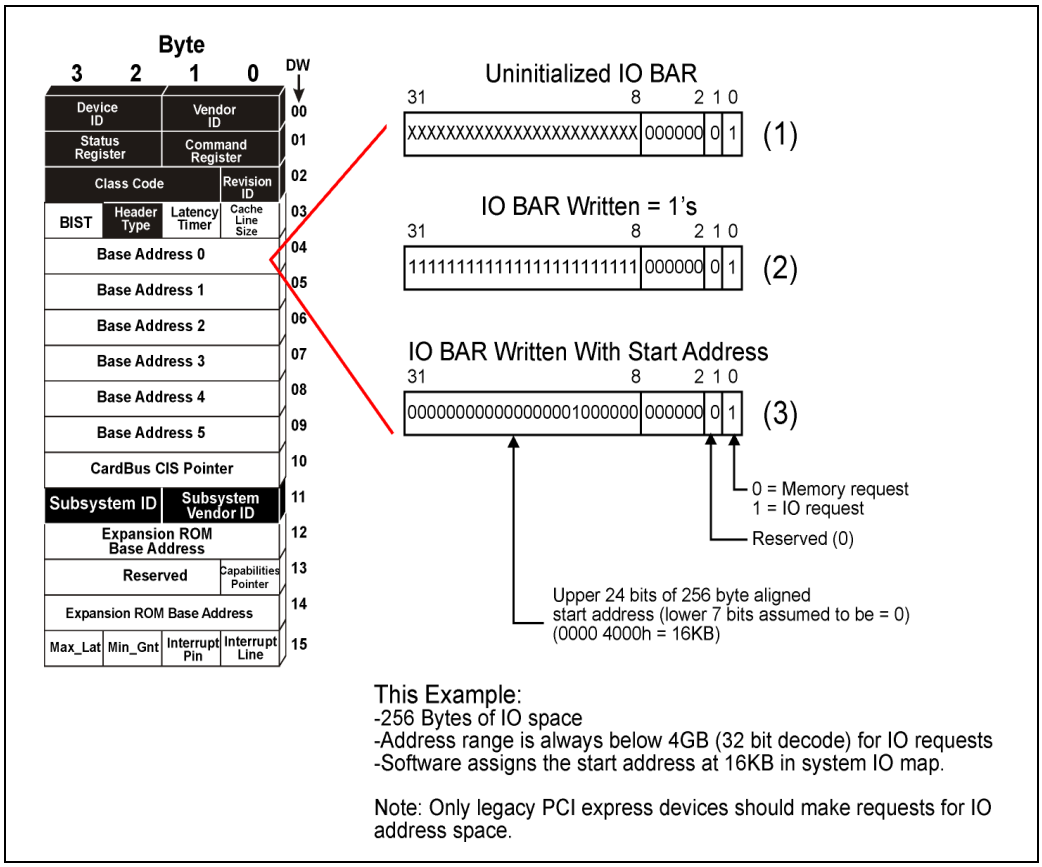


Table 3-10: Results Of Reading The IO BAR after Writing All "1s" To It

BAR Bits	Meaning
0	Read back as a "1", indicating an IO request
1	Reserved. Tied low and read back as "0".
7:2	All read back as "0", used to help indicate the size of the request (also see bit 8)
31:8	All read back as "1" because software has not yet programmed the upper bits with a start address for the block. Note that because bit 8 was the first bit (above bit 1) to read back as written (=1); this indicates the IO request size is 256 bytes (2^8 = 256).

Base/Limit Registers, Type 1 Header Only

General

The second set of configuration registers related to routing are also found in Type 1 configuration headers and used when forwarding address-routed TLPs. Marked "<2" in Figure 3-16 on page 137, these are the three sets of Base/Limit registers programmed in each bridge interface to enable a switch/bridge to claim and forward address-routed TLPs to a secondary bus. Three sets of Base/Limit Registers are needed because transactions are handled differently (e.g. prefetching, write-posting, etc.) in the prefetchable memory, non-prefetchable memory (MMIO), and IO address domains. The Base Register in each pair establishes the start address for the community of downstream devices and the Limit Register defines the upper address for that group of devices. The three sets of Base/Limit Registers include:

- Prefetchable Memory Base and Limit Registers
- Non-Prefetchable Memory Base and Limit Register
- I/O Base and Limit Registers

Prefetchable Memory Base/Limit Registers

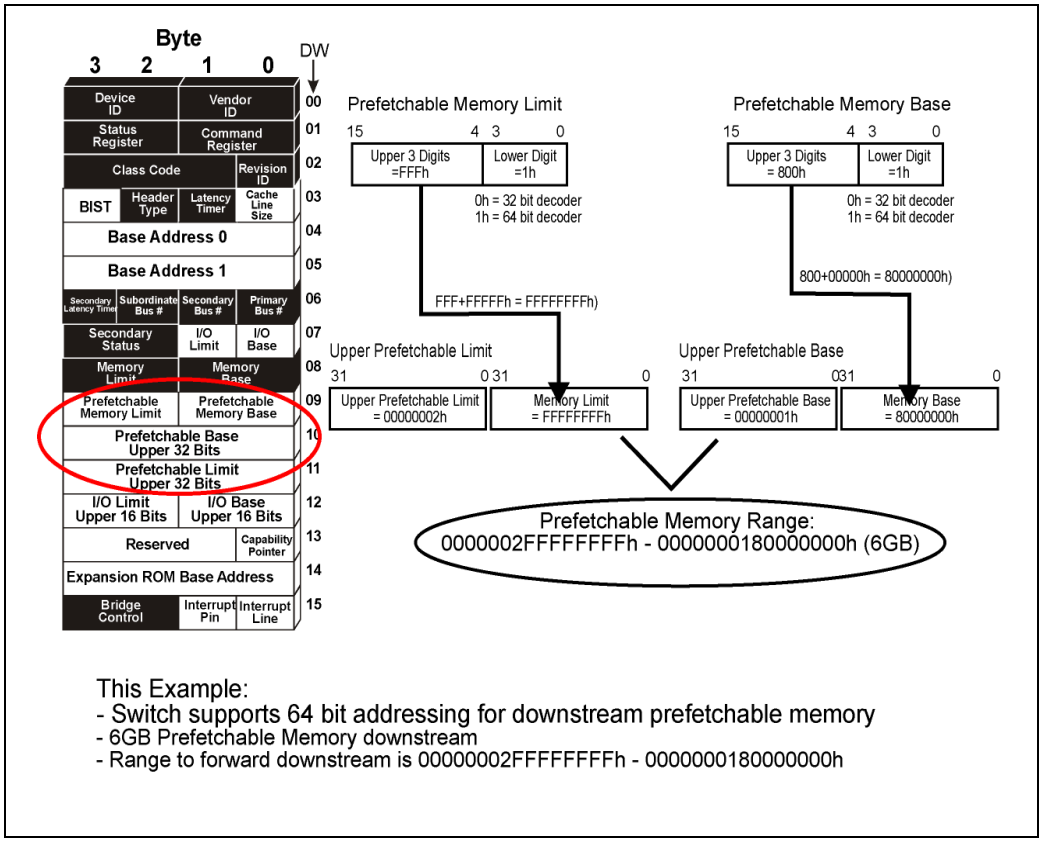
The Prefetchable Memory Base/Limit registers are located at DW 9 and Prefetchable Memory Base/Limit Upper registers at DW 10-11 within the header 1. These registers track all downstream prefetchable memory devices. Either 32 bit or 64 bit addressing can be supported by these registers. If the

Chapter 3: Address Spaces & Transaction Routing

Upper Registers are not implemented, only 32 bits of memory addressing is available, and the TLP headers mapping to this space will be the 3DW format. If the Upper registers and system software maps the device above the 4GB boundary, TLPs accessing the device will carry the 4DW header format. In the example shown in Figure 3-20 on page 145, a 6GB prefetchable address range is being set up for the secondary link of a switch.

Register programming in the example shown in Figure 3-20 on page 145 is summarized in Table 3-11.

Figure 3-20: 6GB, 64-Bit Prefetchable Memory Base/Limit Register Set Up



PCIEX.book Page 146 Tuesday, August 5, 2003 4:22 PM

Table 3-11: 6 GB, 64-Bit Prefetchable Base/Limit Register Setup

Register	Value	Use
Prefetchable Memory Base	8001h	Upper 3 nibbles (800h) are used to provide most significant 3 digits of the 32-bit Base Address for Prefetchable Memory behind this switch. The lower 5 digits of the address are assumed to be 00000h. The least significant nibble of this register value (1h) indicates that a 64 bit address decoder is supported and that the Upper Base/Limit Registers are also used.
Prefetchable Memory Limit	FFF1h	Upper 3 nibbles (FFFh) are used to provide most significant 3 digits of the 32-bit Limit Address for Prefetchable Memory behind this switch. The lower 5 digits of the address are assumed to be FFFFFh. The least significant nibble of this register value (1h) indicates that a 64 bit address decoder is supported and that the Upper Base/Limit Registers are also used.
Prefetchable Memory Base Upper 32 Bits	00000001h	Upper 32 bits of the 64-bit Base address for Prefetchable Memory behind this switch.
Prefetchable Memory Limit Upper 32 Bits	00000002h	Upper 32 bits of the 64-bit Limit address for Prefetchable Memory behind this switch.

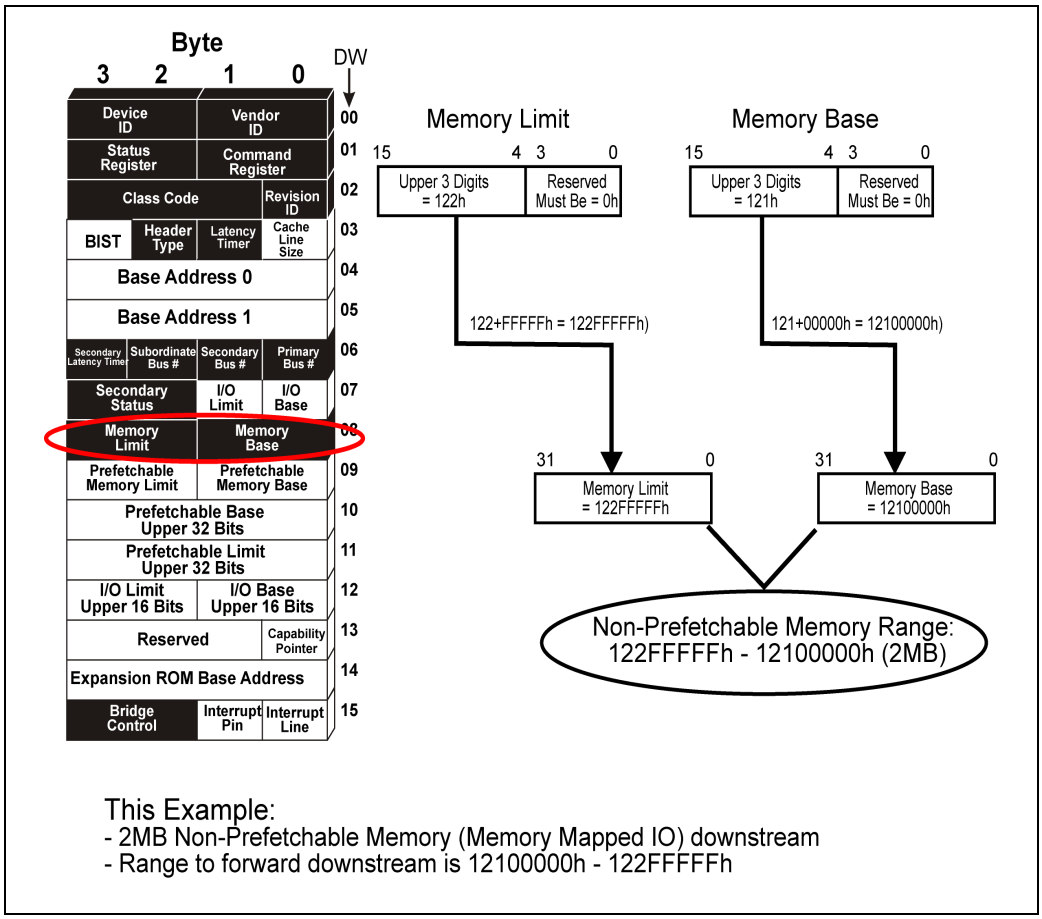
Non-Prefetchable Memory Base/Limit Registers

Non-Prefetchable Memory Base/Limit (at DW 8). These registers are used to track all downstream non-prefetchable memory (memory mapped IO) devices. Non-prefetchable memory devices are limited to 32 bit addressing; TLPs targeting them always use the 3DW header format.

Register programming in the example shown in Figure 3-21 on page 147 is summarized in Table 3-12.

Chapter 3: Address Spaces & Transaction Routing

Figure 3-21: 2MB, 32-Bit Non-Prefetchable Base/Limit Register Set Up



PCIEX.book Page 148 Tuesday, August 5, 2003 4:22 PM

Table 3-12: 2MB, 32-Bit Non-Prefetchable Base/Limit Register Setup

Register	Value	Use
Memory Base (Non-Prefetchable)	1210h	Upper 3 nibbles (121h) are used to provide most significant 3 digits of the 32-bit Base Address for Non-Prefetchable Memory behind this switch. The lower 5 digits of the address are assumed to be 00000h. The least significant nibble of this register value (0h) is reserved and should be set = 0.
Memory Limit (Non-Prefetchable)	1220h	Upper 3 nibbles (122h) are used to provide most significant 3 digits of the 32-bit Limit Address for Prefetchable Memory behind this switch. The lower 5 digits of the address are assumed to be FFFFFh. The least significant nibble of this register value (0h) is reserved and should be set = 0.

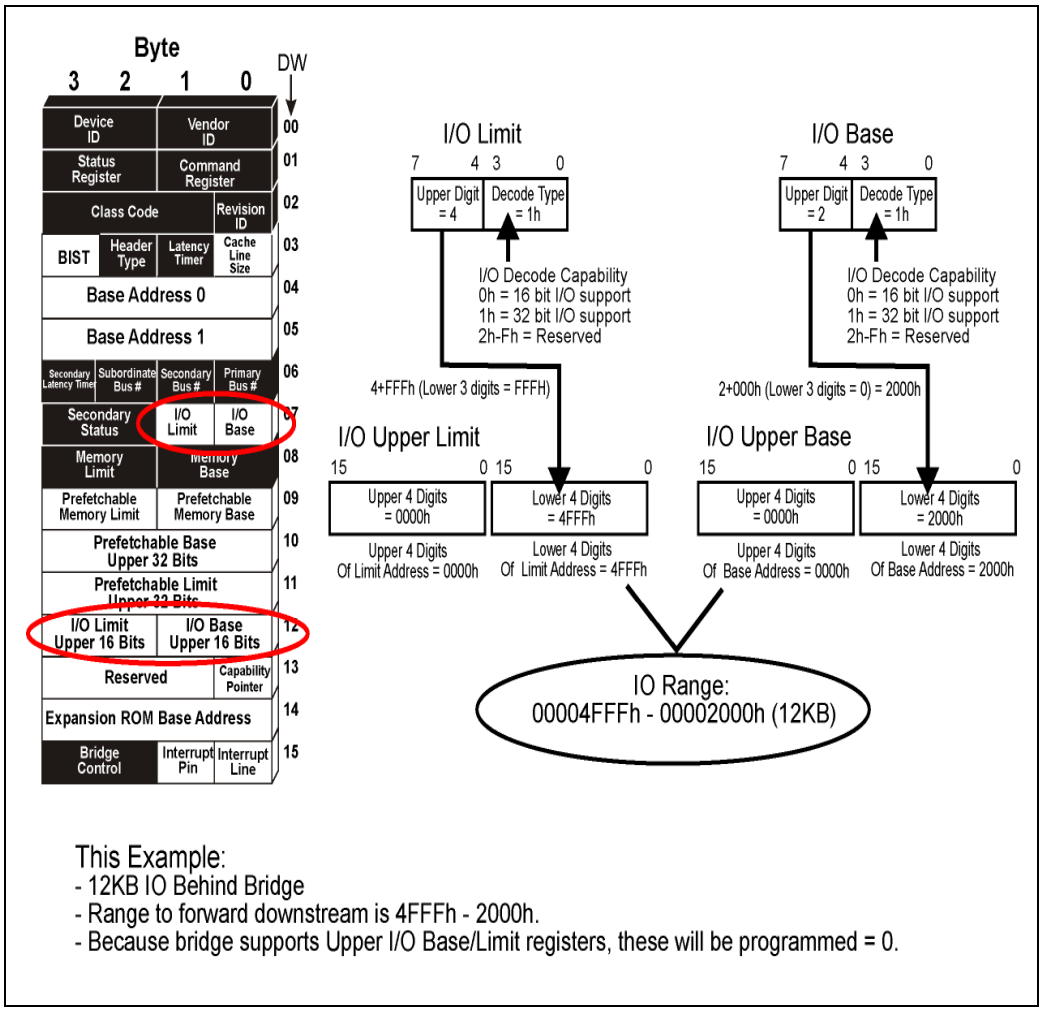
IO Base/Limit Registers

IO Base/Limit (at DW 7) and IO Base/Limit Upper registers (at DW 12). These registers are used to track all downstream IO target devices. If the Upper Registers are used, then IO address space may be extended to a full 32 bits (4GB). If they are not implemented, then IO address space is limited to 16 bits (64KB). In either case, TLPs targeting these IO devices always carry the 3DW header format.

Register programming in the example shown in Figure 3-22 on page 149 is summarized in Table 3-13 on page 150.

PCIEX.book Page 149 Tuesday, August 5, 2003 4:22 PM

Figure 3-22: IO Base/Limit Register Set Up



This Example:
- 12KB IO Behind Bridge
- Range to forward downstream is 4FFFh - 2000h.
- Because bridge supports Upper I/O Base/Limit registers, these will be programmed = 0.

PCIEX.book Page 150 Tuesday, August 5, 2003 4:22 PM

Table 3-13: 256 Byte IO Base/Limit Register Setup

Register	Value	Use
IO Base	21h	Upper nibble (2h) specifies the most significant hex digit of the 32 bit IO Base address (the lower digits are 000h) The lower nibble (1h) indicates that the device supports 32 bit IO behind the bridge interface. This also means the device implements the Upper IO Base/ Limit register set, and those registers will be concatenated with Base/Limit.
IO Limit	41h	Upper nibble (4h) specifies the most significant hex digit of the 32 bit IO Limit address (the lower digits are FFFh). The lower nibble (1h) indicates that the device supports 32 bit IO behind the bridge interface. This also means the device implements the Upper IO Base/ Limit register set, and those registers will be concatenated with Base/Limit.
IO Base Upper 16 Bits	0000h	Upper 16 bits of the 32-bit Base address for IO behind this switch.
IO Limit Upper 16 Bits	0000h	Upper 16 bits of the 32-bit Limit address for IO behind this switch.

Bus Number Registers, Type 1 Header Only

The third set of configuration registers related to routing are used when forwarding ID-routed TLPs, including configuration cycles and completions and optionally messages. These are marked "<3" in Figure 3-16 on page 137. As in PCI, a switch/bridge interface requires three registers: Primary Bus Number, Secondary Bus Number, and Subordinate bus number. The function of these registers is summarized here.

Primary Bus Number

The Primary Bus Number register contains the bus (link) number to which the upstream side of a bridge (switch) is connected. In PCI Express, the primary bus is the one in the direction of the Root Complex and host processor.

Secondary Bus Number

The Secondary Bus Number register contains the bus (link) number to which the downstream side of a bridge (switch) is connected.

Subordinate Bus Number

The Subordinate Bus Number register contains the highest bus (link) number on the downstream side of a bridge (switch). The Subordinate and Secondary Bus Number registers will contain the same value unless there is another bridge (switch) on the secondary side.

A Switch Is a Two-Level Bridge Structure

Because PCI does not natively support bridges with multiple downstream ports, PCI Express switch devices appear logically as two-level PCI bridge structures, consisting of a single bridge to the primary link and an internal PCI bus which hosts one or more virtual bridges to secondary interfaces. Each bridge interface has an independent Type 1 format configuration header with its own sets of Base/Limit Registers and Bus Number Registers. Figure 3-23 on page 152 illustrates the bus numbering associated with the external links and internal bus of a switch. Note that the secondary bus on the primary link interface is the internal virtual bus, and that the primary interface of all downstream link interfaces connect to the internal bus logically.

PCIEX.book Page 152 Tuesday, August 5, 2003 4:22 PM

Figure 3-23: Bus Number Registers In A Switch