PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	Lightweight Notification (LN) Protocol
DATE:	Introduced: Jan 27, 2009; Last Updated Oct 2, 2011 Protocol Workgroup Final Approval: October 6, 2011
AFFECTED DOCUMENT:	PCI Express Base Specification version 3.0
SPONSOR:	Hewlett-Packard, Advanced Micro Devices

Part I

1. Summary of the Functional Changes

This optional normative ECN defines a simple protocol where a device can register interest in one or more cachelines in host memory, and later be notified via a hardware mechanism when any registered cachelines are updated.

The protocol defines LN Reads, LN Completions, and LN Writes, which are special forms of Memory Reads, Completions, and Memory Writes. The striking difference is that LN Reads/Writes result in the Endpoint being notified via the host sending an LN Message when a registered cacheline is updated in the future. The protocol supports two cacheline sizes (CLSs), 64 bytes and 128 bytes. Support for each CLS is optional, both for hosts and for Endpoints. Switches require no functional changes to support LN protocol.

A new LNR Capability structure enables software to discover and manage LN Requester capabilities in Endpoints. A new field in the Device Capabilities 2 register informs software of LN Completer capabilities in the host.

2. Benefits as a Result of the Changes

Some of the potential benefits of LN protocol across PCIe include:

I/O bandwidth and I/O latency reduction: A device that caches data instead of transferring it over PCIe for each access can dramatically reduce I/O latency and consumed I/O bandwidth for some use models. Reducing I/O bandwidth consumption also reduces host memory bandwidth consumption.

Lightweight signaling: A device using LN protocol with host memory can enable host software to signal the device by updating a cacheline instead of performing programmed I/O (PIO) operations, which often incur high software overheads and introduce synchronization and flow-control issues.

Dynamic device associations: Virtual machine (VM) guest drivers communicating with a device strictly via host memory structures (with no PIOs) makes it easier for guests to be migrated, and for VM hosts to switch dynamically between using fully virtualized I/O for that device versus using direct I/O for that device.

3. Assessment of the Impact

There is no impact to systems that do not support LN protocol.

4. Analysis of the Hardware Implications

LN protocol is optional normative. Endpoints can optionally support LN Requester capabilities, which is disabled by default. Root Complexes can optionally support LN Completer capabilities.

Switches require no functional changes to support LN protocol. There is no impact to hardware that does not support it.

5. Analysis of the Software Implications

LN protocol is optional normative. There is no impact to software that does not enable the feature.

6. Analysis of the C&I Test Implications

No existing C&I tests should be impacted unless they test bits, fields, or Extended Capability IDs that were previously Reserved.

Part II

Detailed Description of the change

Modify Terms and Acronyms as shown:

Terms and Acronyms

Lightweight Notification, LN	A lightweight protocol that supports notifications to Endpoints via a hardware mechanism when cachelines of interest are updated.
LN Completer, LNC	A service subsystem in the host that receives LN Read/Write Requests, and sends LN Messages when registered cachelines are updated.
LN Completion	A Completion whose TLP Header has the LN bit Set.
LN Message	An architected Message used for notifications with LN protocol.
LN Read	A Memory Read Request whose TLP Header has the LN bit Set.
LN Requester, LNR	A client subsystem in an Endpoint that sends LN Read/Write Requests and receives LN Messages.
LN Write	A Memory Write Request whose TLP Header has the LN bit Set.

Modify Section 2.1.1.4 as shown:

2.1.1.4. Message Transactions

The Message Transactions, or simply Messages, are used to support in-band communication of events between devices.

In addition to ~~the specified~~ specific Messages defined in this document, PCI Express provides support for vendor-defined Messages using specified Message codes. Except for Vendor-Defined Messages that use the PCI-SIG® Vendor ID (0001h), the ~~The~~ definition of specific vendor-defined Messages is outside the scope of this document.

Modify Section 2.2.1 as shown:

2.2.1. Common Packet Header Fields

…

TC[2:0] – Traffic Class (see Section 2.2.6.6) – bits [6:4] of byte 1

LN – 1b indicates that a Memory Request is an LN Read or LN Write, or that a Completion is an LN Completion.

TH – 1b indicates the presence of TLP Processing Hints (TPH) in the TLP header and optional TPH TLP Prefix (if present) – bit 0 of byte 1 (see Section 2.2.7.1)

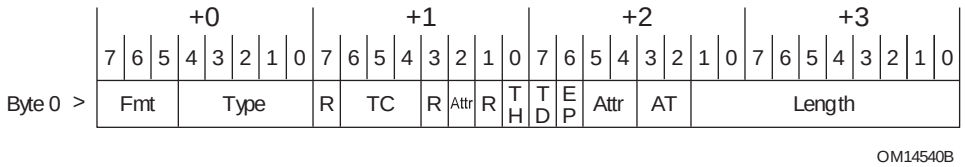

Figure 2-5: Fields Present in All TLP Headers

Note to Editor: modify Figure 2-5 to indicate that bit 1 of byte 1 is now the LN bit. Modify all subsequent TLP Header figures to indicate the same.

Modify Section 2.2.4.1 as shown:

2.2.4.1. Address Based Routing Rules

…

For Memory Read, Memory Write, and AtomicOp Requests, the Address Type (AT) field is encoded as shown in Table 2-5, with full descriptions contained in the *Address Translation Services Specification*. For all other Requests, the AT field is Reserved. LN Reads and LN Writes have special requirements. See Section 6.x.5.

Modify Section 2.2.5 as shown:

2.2.5. First/Last DW Byte Enables Rules

…

- ❑ A Write Request with a length of 1 DW with no bytes enabled is permitted, and has no effect at the Completer unless otherwise specified.

IMPLEMENTATION NOTE

Zero-Length Write

A Memory Write Request of 1 DW with no bytes enabled, or "zero-length Write," may be used by devices under certain protocols, in order to achieve an intended side effect. One example is LN protocol. See Section 6.x.

Modify Section 2.2.7 as shown:

2.2.7. Memory, I/O, and Configuration Request Rules

…

- ❑ I/O Requests have the following restrictions:
  - TC[2:0] must be 000b
  - LN is not applicable to I/O Requests and the bit is Reserved
  - TH is not applicable to I/O Request and the bit is Reserved

- …

❑ Configuration Requests have the following restrictions:

- TC[2:0] must be 000b
- LN is not applicable to Configuration Requests and the bit is Reserved
- TH is not applicable to Configuration Requests and the bit is Reserved

Modify Section 2.2.8 as shown:

2.2.8. Message Request Rules

…

❑ Except as noted, the Attr[1:0] field is Reserved.

❑ LN is not applicable to Message Requests and the bit is Reserved.

❑ Except as noted, TH is not applicable to Message Requests and the bit is Reserved.

Modify Section 2.2.8.6 as shown:

2.2.8.6. Vendor_Defined Messages

The Vendor_Defined Messages allow expansion of PCI Express messaging capabilities, either as a general extension to the PCI Express Specification or a vendor-specific extension. ~~Such extensions are not covered specifically in this document, although future revisions of this specification may use this mechanism to define new Messages (see below).~~ This section defines the rules associated with these Messages generically.

…

2.2.8.6.1. SIG-Defined VDMs

SIG-Defined VDMs are Vendor-Defined Type 1 Messages that use the PCI-SIG® Vendor ID (0001h). As a Vendor-Defined Type 1 Message, each is silently discarded by a Completer if the Completer does not implement it.

Beyond the rules for other Vendor-Defined Type 1 Messages, the following rules apply to the formation of the SIG-Defined VDMs:

❑ SIG-Defined VDMs use the Header format shown in Figure 2.x.

❑ The Requester ID field must contain the value associated with the Requester.

❑ The Message Code must be 01111111b.

❑ The Vendor ID must be 0001h, which is assigned to the PCI-SIG.

❑ The Subtype field distinguishes the specific SIG-Defined VDMs. See Appendix F for a list of SIG-Defined VDMs.

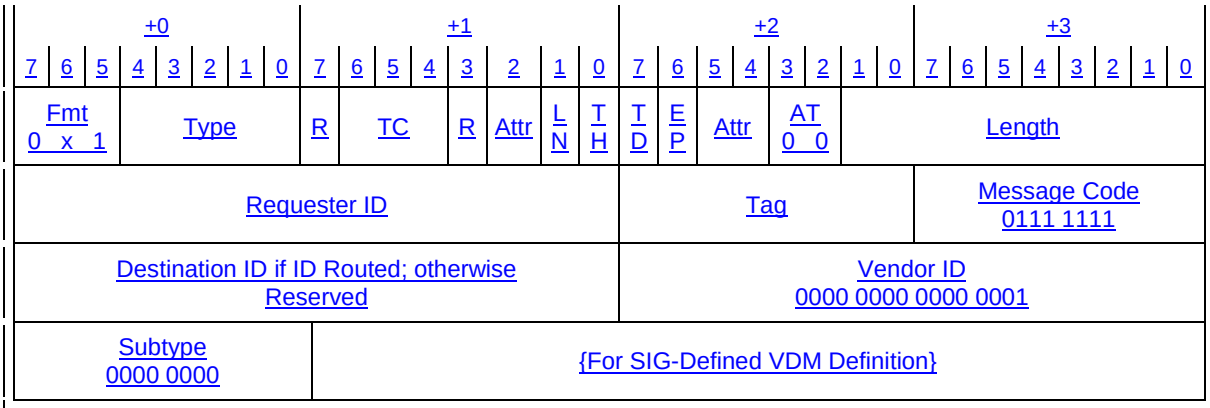

Figure 2-x: Header for SIG-Defined VDMs

2.2.8.6.2. LN Messages

LN protocol (see Section 6.x) defines LN Messages, which are SIG-Defined VDMs. The payload of each Message generally contains the 64-bit Address of a registered cacheline that has been updated or evicted. The single 64-bit address format is used both with 64- and 32-bit addresses. Since each LN Message is a Vendor-Defined Type 1 Message, a Completer that receives a properly formed LN Message is required to silently discard it if the Completer doesn't recognize the Message.

An LN Message can be directed to a single Endpoint using ID-based routing, or broadcast to all devices below a given Root Port. Whether a broadcast LN Message is sent to all Root Ports in the RC is implementation specific.

Beyond the rules for other SIG-Defined VDMs, the following rules apply to the formation of LN Messages:

- ❑ Table 2-x and Figure 2-x define the LN Messages.
- ❑ Each Message must include a 2-DWORD data payload.
- ❑ The Fmt field must be 011b (4 DW Header, with data).
- ❑ The TLP Type must be MsgD.
- ❑ The Length field must be 2.
- ❑ The TC[2:0] field must be 000b.
- ❑ Attr[2], the ID-Based Ordering (IDO) bit, is not Reserved.
- ❑ Attr[1], the Relaxed Ordering (RO) bit, is not Reserved.
- ❑ Attr[0], the No Snoop bit, is Reserved.
- ❑ Note: as with all Messages, the LN bit is Reserved. In contrast, the LN bit must be Set for LN Reads, LN Writes, and LN Completions.
- ❑ The Tag field is Reserved.
- ❑ If the LN Message is the broadcast version, the Destination ID field is Reserved.
- ❑ The Subtype field must be 00h.
- ❑ If the cacheline size in effect for the system is 128 bytes, bit 6 in the Cacheline Address must be Clear. For an LNR receiving an LN Message, if the LNR CLS bit in the LNR

Control register is Set, configuring the LNR for 128-byte cachelines, the LNR must ignore the value of bit 6 in the Cacheline Address.

- The Notification Reason (NR) field is encoded as shown in Table 2-w, indicating the specific reason that the LN Message was sent. These encodings apply to both the directed and broadcast versions of LN Messages.

Table 2-w: Notification Reason (NR) Field Encodings

NR Coding (b)	Description
00	LN Message was sent due to a cacheline update.
01	LN Message was sent due to the eviction of a single cacheline.
10	LN Message was sent due to the eviction of all cachelines registered to this Function. For this case, the Cacheline Address is Reserved.
11	Reserved

Table 2-x: LN Messages

Name	Code[7:0] (b)	Routing[2:0] (b)	Support				Description/Comments
			RC	Ep	Sw	Br	
LN Message (directed)	0111 1111	010	t	r	tr	r	RC directs to a single Endpoint.
LN Message (broadcast)	0111 1111	011	t	r	tr	r	RC broadcasts to all devices under a given Root Port.

The format the LN Message is shown in Figure 2-x below.

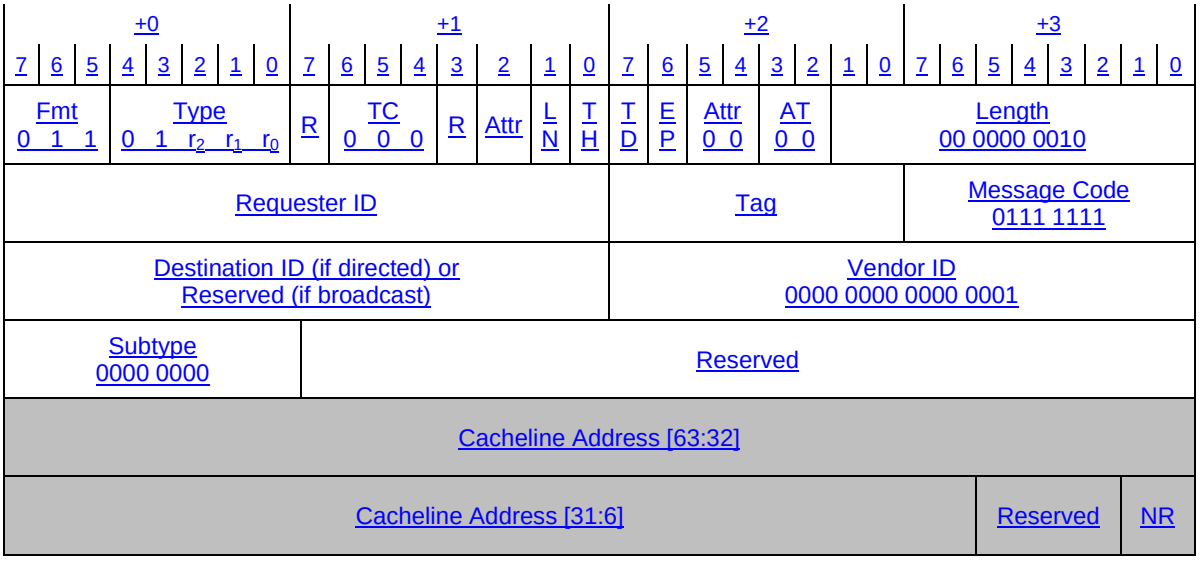

Figure 2-x: LN Message

Modify Section 2.2.9 as shown:

2.2.9. Completion Rules

…

- ❑ Completion headers must supply the same values for the Attribute as were supplied in the header of the corresponding Request, except as explicitly allowed when IDO is used (see Section 2.2.6.4).

- ❑ If the Completer is an LN Completer and the targeted memory region supports registrations, the following rules apply; otherwise the value of the LN Bit must be 0b.
  - If the Completion Status is Successful Completion and the associated Request was an LN Read, the LN bit must be 1b.
  - Otherwise the LN bit must be 0b.

- ❑ the TH bit is reserved for Completions

Add new section in Chapter 6 as follows:

6.x Lightweight Notification (LN) Protocol

Lightweight Notification (LN) protocol enables Endpoints to register interest in specific cachelines in host memory, and be notified via a hardware mechanism when they are updated. An LN Requester (LNR) is a client subsystem in an Endpoint that sends LN Read/Write Requests and receives LN Messages. An LN Completer (LNC) is a service subsystem in the host that receives LN Read/Write Requests, and sends LN Messages when registered cachelines are updated.

LN protocol provides a notification service for when cachelines of interest are updated. Most commonly, an LNR sends an LN Read to a Memory Space range that has an associated LNC, requesting a copy of a cacheline (abbreviated "line" in this section). The LNC returns the requested line to the LNR and records that the LNR has requested notification when that line is updated; that is, the LNC "registers" the line. Later, the LNC notifies the LNR via an LN Message when some entity updates the line, so the LNR can take appropriate action. A similar notification service exists for LN Writes, where an LNR writing a line can be notified later when the line is updated.

LN protocol permits multiple LNRs each to register the same line concurrently. An Endpoint with LNR is permitted to write to a line at any time, regardless of whether the LNR has registered the line.

A typical system consists of host processors, host memory, a host internal fabric, Root Ports, Switches, and Endpoints. Figure 6-x below illustrates a simplified view of the elements and how they are interconnected to provide a context for describing how LN protocol operates.

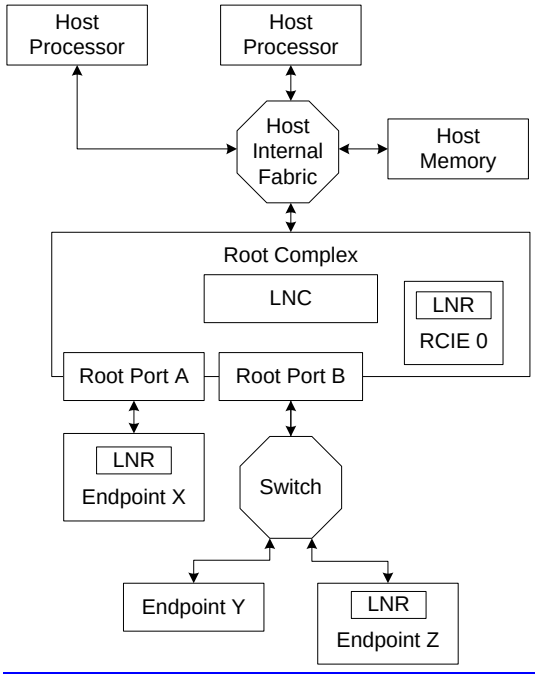

Figure 6-x: Sample System Block Diagram

In the figure above, Endpoint X, Endpoint Z, and RCIE 0 each contain an LNR. The Root Complex contains an LNC.

6.x.1 LN Protocol Operation

LN is a simple protocol that satisfies several key use models with minimum complexity and cost.

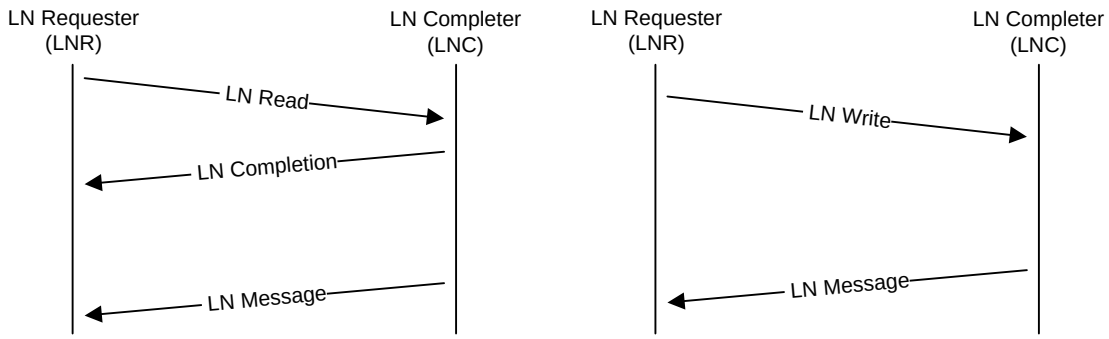

Figure 6-x: LN Protocol Basic Operation

LN protocol operation with a single LNR is shown in the diagram above. For the case of reads, as shown on the left: (1) an LNR requests a copy of a line from host memory using an LN Read; (2) the LNC returns the line in an LN Completion and records that the LNR has registered the line; and (3) the LNC later notifies the LNR using an LN Message when the registered line has been updated. For the case of writes, as shown on the right: (1) an LNR writes to a line to host memory using an LN Write; (2) the LNC records that the LNR has registered the line; and (3) the LNC later notifies the LNR using an LN Message when the registered line has been updated.

An LNC must send an LN Message either if a registered line is updated by some entity (e.g., a CPU or a device) or if the LNC is no longer going to track the state of that line. The latter case is referred to as an eviction, and is indicated by the Notification Reason (NR) field in the LN Message.

If an LN Requester does an LN Read or LN Write to a line that it already has registered, then the LN Requester generally can't determine if a subsequent LN Message it receives for that line was for the most recent LN Read/Write Request or for a previous one.

LN protocol permits multiple LNRs to register the same line concurrently. In this case the LNC notifies the multiple LNRs either by sending a directed LN Message to each LNR, or by sending a broadcast LN Message to each affected Root Port hierarchy.

Once the LNC updates or evicts a given line and sends one or more LN Messages to notify all LNRs with associated registrations, the LNC will not send additional LN Messages for that specific line unless it receives a new LN Read or LN Write to that line.

An LNC is permitted to have an implementation-specific limit on how many LNRs it can independently track for each given line. At or below this limit, the LNC generally uses directed LN Messages for notifications. Above this limit, the LNC uses broadcast LN Messages. An implementation is permitted to have a limit of zero, and use broadcast LN Messages exclusively.

A single LN Message with a specific NR field value can indicate that all lines registered to that LNR have been evicted. Both directed and broadcast versions of this "all evicted" LN Message are permitted.

An LNC is permitted to have an implementation-specific limitations on how many lines or sets of lines it can register concurrently. If the LNC receives a Request to register a new line, but the LNC has insufficient resources to do so, the LNC is permitted to evict one or more old lines in order to free up the necessary resources, or send a LN Message indicating that the new line was evicted.

IMPLEMENTATION NOTE

Excessive Use of Broadcast LN Messages

In order to avoid performance issues, LNCs that use broadcast LN Messages should be implemented to minimize the number of RP hierarchies each broadcast LN Message is sent to, and also to keep the rate of broadcast LN Messages within reasonable bounds. Each broadcast LN Message consumes Link bandwidth, and some Endpoints may process broadcast LN Messages at a relatively low rate.

In particular, Endpoints that do not support LN Requester capability may handle received broadcast LN Messages as an exception case using a low performance mechanism, e.g., decoding the Message with device firmware instead of in hardware. Each Message could conceivably take microseconds to process[1], and an excessive rate of them could easily cause back-pressure of Posted Requests within the fabric, causing severe performance problems.

[1] The Posted Request Acceptance Limit permits an Endpoint to take up to 10 µs to process each received Posted Request. See Section 2.3.1.

While Endpoints that do not support LN Requester capability may also handle directed LN Messages as an exception case using a low performance mechanism, this should not cause performance problems. With LN protocol, LNCs send directed LN Messages only to Endpoints that support LN Requester capability, and presumably those Endpoints will be able to process both directed and broadcast LN Messages at a rate that avoids performance issues.

6.x.2 LN Registration Management

Since LNCs have limited resources for registering cachelines, an LNR Capability structure in each LNR provides mechanisms for software to limit the LNR's maximum number of outstanding registrations. Software reads the LNR Registration Max field to discover the maximum number an LNR is capable of having outstanding. If necessary, software can set the LNR Registration Limit field to impose a specified limit.

LNC registration resource limitations may be highly implementation specific, perhaps including factors such as set associativity, maximum number of sets, and distinct sets of resources for different regions of host memory. How software discovers these resource limitations is outside the scope of this specification.

To manage its number of outstanding registrations, an LNR can deregister outstanding registrations by sending a zero-length LN Write to each line it wants to deregister.

The LNC is not required to accept registrations for all locations in host memory. However, the granularity for memory regions accepting registrations is required to be no finer than aligned 4KB regions. To determine if a given aligned 4KB region accepts registrations, an LNR can perform an LN Read to any cacheline within the region, and see if an LN Completion is returned. If an LNR wishes to "probe" a region for registration capability without actually creating a registration, the LNR can use a zero-length LN Read, which is required to provide this semantic.

6.x.3 LN Ordering Considerations

LN Reads have the same ordering requirements as other Memory Read Requests, and LN Writes have the same ordering requirements as other Memory Write Requests. LN Completions (for LN Reads) have the same ordering requirements as Completions for other Memory Reads. LN Messages have the same ordering requirements as other Posted Requests.

For a given line, when an LN Completer receives an LN Read followed by an update to that line, the LN Completer is permitted to send the LN Completion and the LN Message in either order.

For a given line, when an LN Completer receives an LN Read that triggers an eviction LN Message, the LN Completer is permitted to send the LN Completion and the LN Message in either order.

For a given line, when an LN Completer receives an LN Write that triggers an eviction LN Message, followed by an update to that line, the LN Completer is permitted to send the two LN Messages in either order.

For different lines, an LN Completer is permitted to send LN Messages in any order.

6.x.4 LN Software Configuration

LN protocol supports 2 cacheline sizes (CLSs) – 64 bytes and 128 bytes. Support for each CLS is optional, both for Root Complexes and Endpoints. The CLS in effect for the system is determined by the host, and indicated by the LN System CLS field in applicable Root Ports and RCRBs. All Root Ports and RCRBs that indicate LN Completer support must indicate the same CLS; otherwise, the results are undefined. The host must not change the system CLS while any operating system is running; otherwise, the results are undefined.

Endpoints supporting LN protocol must support one or both CLSs, and indicate this via the LNR-64 Supported and LNR-128 Supported capability bits. When enabling each LN Requester, software must ensure that the associated LNR CLS control bit is configured to match the system CLS; otherwise, the results are undefined. An LN Requester that supports only one CLS is permitted to hardwire its LNR CLS control bit to the corresponding value.

Software must not change the value of the LNR CLS control bit or the LNR Registration Limit field unless its LNR Enable control bit is already Clear or is being Cleared with the same Configuration Write; otherwise, the results are undefined. If the LNR Enable bit is already Clear, software is permitted to change the values of the LNR CLS bit and LNR Registration Limit field concurrently with Setting the LNR Enable bit, using a single Configuration Write.

Software is permitted to Clear the LNR Enable bit at any time. When the LNR Enable bit is Clear, the LNR must clear all its internal registration state.

6.x.5 LN Protocol Summary

Detailed rules and requirements for LN protocol are distributed throughout the rest of this specification, but here is a general summary plus some unique requirements.

- ❑ An LN Read is a Memory Read Request whose TLP Header has the LN bit Set. An LN Completion is a Completion whose TLP Header has the LN bit Set. An LN Write is a Memory Write Request whose TLP Header has the LN bit Set.
- ❑ All requirements for Memory Read Requests apply to LN Reads unless explicitly stated otherwise.
  - An LN Read must access no more than a single cacheline, as determined by the system CLS. An LN Completer must handle a violation of this rule as a Completer Abort unless the Completer detects a higher precedence error. Partial cacheline LN Reads, including zero-length LN Reads, are permitted.
  - If an LN Completer handles an LN Read as an Uncorrectable Error or an Advisory Non-Fatal Error, the LN Completer must not register notification service for that Request.
  - An LN Completer must handle a zero-length LN Read as a "probe" of the targeted memory region for registration capability. See Section 6.x.2. If the Completion Status is Successful Completion, the LN bit in the TLP Header must indicate if the region supports registration capability, but the LNC must not create a registration for this case. LN Completers must support registration capability with a granularity no finer than aligned 4KB regions.

- Ordering and Flow Control rules for LN Reads are identical to those for Memory Read Requests.

❑ All requirements for Memory Read Completions apply to LN Completions unless explicitly stated otherwise.

- The Completion for an LN Read must be an LN Completion (i.e., have the LN bit Set in its TLP Header) if the Completer is an LN Completer, the targeted memory region accepts registrations, and the Completion Status is Successful Completion; otherwise the Completion must have the LN bit Clear in its TLP Header. Note that a poisoned Completion will have a Completion Status of Successful Completion. See Section 2.7.2.2.

- Ordering and Flow Control rules for LN Completions are identical to Completions whose TLP Header has the LN bit Clear.

❑ All requirements for Memory Write Requests apply to LN Writes unless explicitly stated otherwise.

- An LN Write must access no more than a single cacheline, as determined by the system CLS. An LN Completer must handle a violation of this rule as a Completer Abort unless the Completer detects a higher precedence error.

- If an LN Completer handles an LN Write as an Uncorrectable Error, the LN Completer must not register notification service for that Request. Note that depending upon its configuration, a Completer may handle a poisoned LN Write as an Uncorrectable Error, an Advisory Non-Fatal Error, or a masked error.

- An LN Completer must handle a zero-length LN Write as a request to deregister an existing registration. See Section 6.x.2. If the cacheline targeted by a zero-length LN Write was not previously registered, it must remain unregistered.

- Ordering, Flow Control, and Data Poisoning rules for LN Writes are identical to those for Memory Write Requests.

- A Requester must not generate LN Writes for MSI or MSI-X interrupts. An LN Completer must handle an LN Write targeting an interrupt address range as a Completer Abort unless the Completer detects a higher precedence error.

❑ For LN Reads and LN Writes, the address must be the proper type, as indicated by the Address Type (AT) field. See Section 2.2.4.1. The proper type depends upon whether a Translation Agent (TA) is being used. See the *Address Translation Services (ATS) Specification*.

- If a TA is being used, the address must be a Translated address. An LN Requester must support ATS in order to acquire and use Translated addresses.

- If a TA is not being used, the address must be a Default/Untranslated Address.

- An LN Completer detecting a violation of the above rules must handle the Request as a Completer Abort (CA), unless a higher precedence error is detected.

Modify Section 7.8.15 as shown:

7.8.15 Device Capabilities 2 Register (Offset 24h)

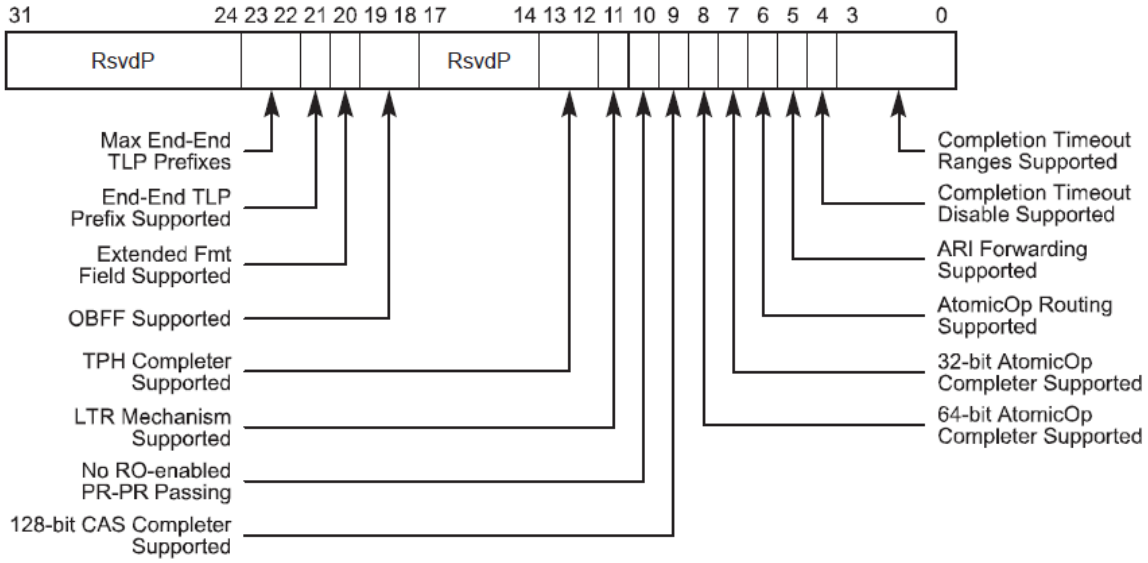

Figure 7-25: Device Capabilities 2 Register

Note to Editor: modify Figure 7-25 to reflect the new field defined below:

Table 7-24: Device Capabilities 2 Register

Bit Location	Register Description	Attributes
…	…	…
15:14	LN System CLS – Applicable only to Root Ports and RCRBs; must be 00b for all other Function types. This field indicates if the Root Port or RCRB supports LN protocol as an LN Completer, and if so, what cacheline size is in effect. Encodings are: 00b LN Completer either not supported or not in effect 01b LN Completer with 64-byte cachelines in effect 10b LN Completer with 128-byte cachelines in effect 11b Reserved	HwInit

Add the following section to Chapter 7:

7.x. LNR Extended Capability

The LN Requester (LNR) Extended Capability is an optional normative capability for Endpoints. All Endpoints that support LN protocol as a Requester must implement this capability. See Section 6.x. This capability may be implemented by any type of Endpoint, but not by any other Function type.

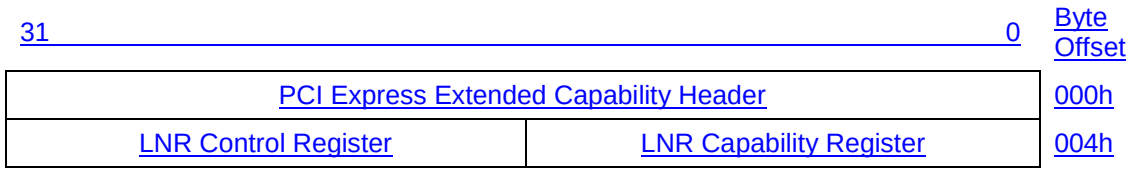

Figure 7-xx LNR Extended Capability

7.x.1. LNR Extended Capability Header (Offset 00h)

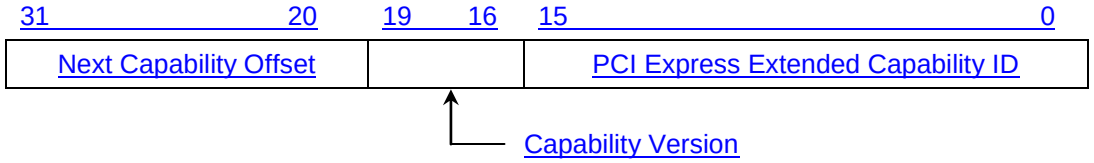

Figure 7-xx LNR Extended Capability Header

Table 7-xx LNR Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the extended capability. PCI Express Extended Capability ID for the LNR Extended Capability is 001Ch.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Extended Capability structure or 000h if no other items exist in the linked list of capabilities.	RO

7.x.2. LNR Capability Register (Offset 04h)

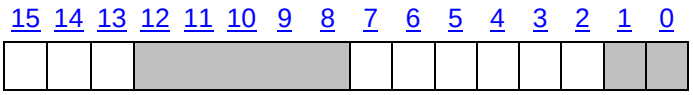

Figure 7-xx LNR Capability Register

Note to Editor: modify the above figure to indicate the shaded bits and fields, which are defined in the table below. All other bits are RsvdP.

Table 7-xx LNR Capability Register

Bit Location	Register Description	Attributes
0	LNR-64 Supported – This bit must be 1b if the Endpoint supports LN protocol for 64-byte cachelines as a Requester; otherwise, must be 0b. See Section 6.x.4 for additional details.	RO
1	LNR-128 Supported – This bit must be 1b if the Endpoint supports LN protocol for 128-byte cachelines as a Requester; otherwise, must be 0b.	RO
12:8	LNR Registration Max – This field, encoded as a power of 2, indicates the maximum number of cachelines that this LN Requester is capable of registering concurrently. For example, a value of 00101b indicates that the LN Requester might be capable of registering up to 32 cachelines (2^5) concurrently, and is capable of registering greater than 16.	RO

7.x.3. LNR Control Register (Offset 04h)

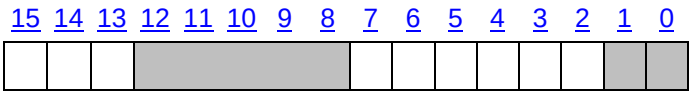

Figure 7-xx LNR Control Register

Note to Editor: modify the above figure to indicate the shaded bits and fields, which are defined in the table below. All other bits are RsvdP.

Table 7-xx LNR Control Register

Bit Location	Register Description	Attributes
0	LNR Enable – When this bit is Set, the Endpoint is enabled to operate as an LN Requester. Software is permitted to Clear this bit at any time. See Section 6.x.4 for requirements regarding the LNR's internal registration state. Default value of this bit is 0b.	RW
1	LNR CLS – This bit controls or indicates the cacheline size used with LN protocol by this Requester. See Section 6.x.4 for restrictions on setting and modifying this bit. If this bit is Clear, the cacheline size is 64 bytes. If this bit is Set, the cacheline size is 128 bytes. If this LN Requester supports only one cacheline size, this bit is permitted to be hardwired to indicate that size. Otherwise, the default value of this bit is 0b.	RW
12:8	LNR Registration Limit – This field, encoded as a power of 2, imposes a limit on the number of cachelines that this LN Requester is permitted to register concurrently. For example, a value of 00100b indicates that the LN Requester must not register more than 16 cachelines (2^4) concurrently. See Section 6.x.4 for restrictions on modifying this field. The default value of this field is 11111b.	RW

Modify Appendix F as shown:

F. Message Code Usage

~~This appendix~~Table F-1 contains a list of currently defined PCI Express Message Codes. Message codes are defined in this specification and in other specifications. This table will be updated as Messages are defined in other specifications but due to document release schedules, this table ~~may~~ might not contain recently defined Messages.

Table F-1: Message Code Usage

…

Table F-2 contains a list of currently defined Subtype codes for SIG-Defined VDMs (see Section 2.2.8.6.1). Subtype codes are defined in this specification and in other specifications. This table will be updated as Subtype codes are defined in other specifications but due to document release schedules, this table might not contain recently defined Subtypes.

Table F-2: SIG-Defined VDM Subtype Usage

Subtype	Routing r[2:0]	Type	Description
0000 0000	010 or 011	MsgD	LN Message, see Section 2.2.8.6.1.1