* 读取目标位置的值。
* 将“swap”值写
* 返回目标位置的原始值
* CAS（Compare and Swap）：请求包含两个操作数，一个
  + 读取目标位置的值。
  + 将该值与“比较”值进行
  + 如果相等，则将“swap”值写
  + 返回目标位置的原始值

给定的AtomicOp事务具有关联的操作数大小，并且相同的大小用于目标位置访问和返回值。FetchAdd和Swap支持32位和64位的操作数大小CAS支持32、64和128位的操作数大小

AtomicOp功能是可选的规范。允许端点和根端口实现AtomicOptener功能。允许具有内存空间BAR的PCI Express功能以及所有根端口实现AtomicOp完成器功能。路由元件（交换机以及支持根端口之间的对等访问的根复合体）需要AtomicOp路由能力以便路由AtomicOp请求。AtomicOps是为设备到主机、设备到设备和主机到设备事务而设计的在每种情况下，路由器、完成器和所有中间路由元素都必须支持相关的AtomicOp功能。

PCI Express到PCI/PCI-X桥接器不支持AtomicOp功能如果需要的话，锁定事务可以用于这样的网桥之下的设备。AtomicOps和Locked Transactions可以在同一层次结构上并发操作。

软件通过设备功能2寄存器中的三个新位发现特定的AtomicOp完成器功能（参见

[部分7.5.3.15](#_bookmark13)）。为了提高互操作性，需要根端口来实现某些AtomicOp Completer功能（如果有的话）（请参见[6.15.3.1部分](#_bookmark120)）。软件通过设备功能2寄存器中的AtomicOp路由支持位AtomicOp编译器功能的软件发现超出了本规范的范围，但软件必须在功能的设备控制2寄存器中设置AtomicOp编译器启用位，[7.5.3.16](#_bookmark0)

通过路由元素，软件可以在逐个端口的基础上设置AtomicOp出口阻塞位（参见[7.5.3.16部分](#_bookmark0)），以避免AtomicOp请求被转发到不应该接收它们的组件每个被阻止的请求都作为AtomicOp Egress Blocked错误处理，默认情况下是Advisory Non-Fatal Error。

AtomicOps是存储器事务，因此用于管理存储器空间访问的现有标准机制（例如，总线主机使能、内存空间使能和基址寄存器）。

* + 1. AtomicOp使用模式和好处

AtomicOps支持高级同步机制，当有多个生产者和/或多个消费者需要以非阻塞方式同步时，这种机制例如，多个生产者可以安全地排队到一个公共队列，而无需任何显式锁定。

AtomicOps还支持无锁统计计数器，例如，设备可以自动递增计数器，主机软件可以自动读取和清除计数器。

通过PCI Express直接支持三种选定的AtomicOps，可以更轻松地将现有的高性能SMP应用程序迁移到使用PCI Express作为紧密耦合加速器、协处理器或GP-GPU互连的系统例如，使用PCI PCI附加加速器的移植应用程序可能能够使用与早期SMP应用程序相同的同步算法和数据结构

对给定目标的AtomicOp通常会引起与对同一目标的内存读取相当的延迟在单个层次结构中，多个AtomicOps可以AtomicOps通常会对其他PCI Express流量造成可忽略不计的中断

与Locked Transactions相比，AtomicOps提供更低的延迟、更高的可扩展性、高级的同步算法，并且对其他PCI Express流量的影响大大降低

* + 1. AtomicOp事务协议摘要

AtomicOps的详细协议规则和要求分布在本规范的其余部分，但这里是一个简短的总结和一些独特的要求。

* + - * AtomicOps是Non-Posted Memory Transactions，支持32位和64位地址格式。
      * FetchAdd、Swap和CAS各自使用不同的类型代码。
      * 完成器从长度字段值和AtomicOp请求中的类型代码推断操作数大小
      * AtomicOp完成器用于在目标位置读取和写入数据的字节序格式是特定于实现的，并且允许是完成器确定适合于目标存储器的任何格式（例如，little-endian、big-endian等）。参见[第2.2.2](#_bookmark29)。
      * 如果AtomicOp编译器支持地址转换服务（ATS），则只有当转换后的地址具有适当的访问权限时，才允许编译器在AtomicOp请求中使用转换后的地址具体来说，读取（R）和写入（W）字段都必须设置，并且仅限未翻译访问（U）字段必须清除。参见[2.2.4.1部分](#_bookmark79)。
      * 如果支持访问控制服务（ACS）的组件支持AtomicOp路由或AtomicOp路由器能力，则它处理AtomicOp请求和完成与关于ACS功能的其他存储器请求和完成相同。
      * [NoSnoop](#_bookmark94)属性是适用的，并且允许使用AtomicOp请求进行设置，但是无论[NoSnoop](#_bookmark94)属性值如何，都必须保证原子性
      * [Relaxed Ordering](#_bookmark0)属性是适用的，并且允许与AtomicOp请求一起设置，其中它影响请求及其相关联的完成的顺序
      * AtomicOp请求的排序要求与非发布写入请求的排序要求类似因此，如果验证者想要确保AtomicOp请求在随后的发布或非发布请求之前被完成者观察到，则验证者必须在发出随后的请求之前等待AtomicOp完成
      * AtomicOp Completion的排序要求与Read Completion类似
      * 除非存在参见[2.7.2.2部分](#_bookmark0)。目标位置的值必须保持不变。
      * 如果AtomicOp请求的完成者在访问目标位置或执行Atomic操作时遇到无法纠正的错误，完成者必须将其作为完成者中止（CA）处理。目标位置的后续状态是特定于实现的
      * AtomicOp-aware Completers需要处理任何正确形成的AtomicOp请求，其类型或操作数大小如果AtomicOp请求中的Length字段包含未架构的值，则该请求必须由AtomicOp感知的完成程序作为格式错误的TLP处理。参见[第2.2.7](#_bookmark80)。
      * 如果[多功能设备](#_bookmark116)中的任何功能支持AtomicOp完成器或AtomicOp路由能力，则该设备中具有存储器空间BAR的所有功能必须解码正确形成的AtomicOp请求，并将它们不支持的任何请求注意，在这种装置中，功能缺乏

禁止AtomicOp完成器功能将正确形成的AtomicOp请求作为格式错误的TLP处理。

* + - * 如果RC具有支持AtomicOp路由能力的任何根端口，则RC中通过转发的AtomicOp请求可到达的所有RCiEP必须解码正确形成的AtomicOp请求，并将它们不支持的任何请求
      * 对于具有受支持的类型和操作数大小的AtomicOp请求，需要AtomicOp感知完成器执行请求或将其作为完成器中止（CA）处理其目标内存空间中的任何位置。允许完成者根据其编程模型的需要在其目标内存空间的子集上支持AtomicOp请求（参见[第2.3.1节](#_bookmark95)）。由PCI Express定义或继承的存储器空间结构（例如，MSI-X表结构）不需要被支持为AtomicOp目标，除非在结构的描述中明确地声明
      * 对于交换机或RC，当在出口端口中启用了AtomicOp出口阻塞，并且AtomicOp请求的目标是离开该出口端口时，出口端口必须将请求作为AtomicOp出口阻塞错误127（参见[图6-2](#_bookmark3)）处理，并且还必须返回完成状态为UR的完成如果AtomicOp Egress Blocked错误的严重性是非致命的，则必须将这种情况作为[第6.2.3.2.4.1节](#_bookmark106)中所述的咨询性非致命错误进行处理。
    1. AtomicOps的根复合体支持

RC对AtomicOp功能有独特的要求和考虑

* + - 1. 具有AtomicOp完成程序功能的根端口

用于根端口的AtomicOp完成器能力指示根端口支持在其入口端口处接收以由根端口BAR分配的主机存储器或存储器空间为目标的AtomicOp请求这独立于任何具有AtomicOp完成程序功能的

如果根端口为主机内存访问实现任何AtomicOp完成器功能，则它必须实现所有32位和64位AtomicOp完成器功能。实现128位CAS完成程序功能是可选的。

如果RC具有实现AtomicOp完成器能力的一个或多个根端口，则RC必须确保代表给定AtomicOp请求对目标位置的主机存储器访问相对于对该目标位置范围的每个主机处理器或设备访问原子地执行

如果主机处理器通过其指令集架构支持原子操作，则RC还必须确保代表给定AtomicOp请求的主机存储器访问保持任何主机处理器原子操作的原子性。

* + - 1. 具有AtomicOp路由功能的根端口

与其他PCI Express事务一样，对AtomicOp请求和根端口之间完成的对等路由的支持是可选的，并且依赖于实现如果RC支持两个或多个根端口之间的AtomicOp路由功能，则必须通过设备功能2寄存器中的AtomicOp路由支持位

1. 虽然AtomicOp Egress Blocked错误通过返回带有UR Status的Completion来处理，但该错误不会作为Unsupported Request处理例如，它不会设置设备状态寄存器中的“检测到不支持的请求”位

RC不需要支持具有AtomicOp Routing Supported位设置的所有根端口对之间的AtomicOp路由需要在不支持的根端口对之间路由的AtomicOp请求必须作为不支持的请求（UR）处理，并由“发送”端口报告

对于支持转发由主机软件或RCiEP发起的AtomicOp请求的任何根端口，必须设置AtomicOpRouting Supported位对于支持将在其入口端口上接收到的AtomicOp请求转发到RCiEP的任何根端口，必须设置AtomicOpRouting Supported位

* + - 1. 具有AtomicOpter功能的RC

允许RC实现主机软件或RCiEP发起AtomicOp请求的能力AtomicOp编译器功能的软件发现超出了本规范的范围

如果RC支持软件启动的AtomicOp请求器功能，则在主机处理器上运行的软件如何使RC生成AtomicOp请求的特定机制不在本规范的范围内

执行说明

通过主机处理器软件生成AtomicOp请求

如果主机处理器指令集架构（ISA）支持直接对应于一个或多个PCI Express AtomicOps的原子操作指令，则RC可以处理以PCI Express存储器空间为目标的相关联的内部原子事务，就像它处理由处理器加载指令产生的内部读取事务然而，RC将内部原子事务导出为PCI Express AtomicOp请求，而不是将即使RC对某些AtomicOp类型和操作数大小使用“导出”方法，它也不需要对所有类型和操作数大小

对于RC不使用“导出”方法的AtomicOp类型和操作数大小，RC可能使用基于RC寄存器的机制，类似于某些PCI主桥使用CONFIG\_ADDRESS和CONFIG\_DATA寄存器来生成配置请求的详情请参阅[[PCI](#_bookmark8)]

“导出”方法可以允许大量并发的AtomicOp请求，使用这种方法也可以更容易地支持从用户空间软件生成AtomicOp请求

基于RC寄存器的机制提供了适用于所有AtomicOp类型和操作数大小的优势，即使主机处理器ISA它还可能支持等待AtomicOp完成的轮询模式，而不是在等待完成时停止处理器

* + 1. 交换机对AtomicOps的

如果一个交换机支持AtomicOp路由功能的任何端口，它必须这样做的所有

* 1. 动态功率分配（DPA）能力

管理功耗的常见方法是通过设备驱动程序、操作系统和执行应用程序之间的协商为这样的设备添加动态功率分配预期作为该协商的扩展通过本规范范围之外的软件机制来完成一些

设备不具有设备专用驱动器来有效地管理功率DPA能力提供了一种为这些类型的设备动态分配功率的机制DPA是适用于端点功能的可选规范功能如果支持，紧急功率降低状态将覆盖此处列出的机制（参见[第6.25](#_bookmark142)）。

DPA功能使软件能够在[D0](#_bookmark36)状态下主动管理和优化功能功耗DPA不适用于电源状态[D1](#_bookmark62)-[D3](#_bookmark37)，因此DPA功能独立于PCI-PM功能进行管理。

DPA定义了一组功率子状态，每个功率子状态具有相关联的功率分配。多达32个子状态[0..[31]可以定义为函数。子状态0是默认的子状态，表示功能能够消耗的最大功率

子状态必须从0到Substate\_Max连续编号，如[第7.9.12.2](#_bookmark23)中所定义。每个连续子状态具有低于或等于先前子状态的功率分配例如，具有四个子状态的函数可以定义如下：

1. 子状态0（默认值）定义25瓦的功率分配。
2. 子状态1定义20瓦的功率分配。
3. 子状态2定义了20瓦的功率分配。
4. 子状态3定义了10瓦的功率分配。

当函数被初始化时，它将在与子状态0相关联的功率分配内操作软件不需要通过中间子状态进行随着时间的推移，软件可以动态地配置功能以在其选择的任何序列中的任何子状态在功能完成先前启动的子状态转换之前，允许软件将功能配置为在任何子状态下操作

子状态转换完成后，函数必须将其子状态与配置的子状态进行比较。如果功能子状态与配置子状态不匹配，则功能必须开始转换到配置子状态。允许功能在配置请求上动态改变子状态转换，指示功能在新的子状态下操作

在先前的示例中，软件可以将功能配置为转换到子状态4，然后是子状态1，然后是子状态3，等等。因此，当软件配置相关控制字段时，该函数必须能够在任何子状态之间转换。

Substate Control Enabled（子状态控制启用）位提供一种机制，允许DPA能力与上述软件协商机制结合使用设置时，功率分配由DPA能力控制清除时，DPA功能被禁用，并且不允许该功能根据子状态控制寄存器字段的配置直接启动子状态在适当的时间点，参与上述软件协商机制的软件清除该位，有效地接管对该功能的功率分配的控制

要求功能在任何子状态下响应配置空间访问

在任何时刻，功能都不得消耗超过其子状态状态指示的功率。当功能被配置为从较高功率子状态转换到较低功率子状态时，功能的子状态状态必须在当功能被配置为从较低功率子状态转换到较高功率子状态时，功能

由于应用的多样性和给定函数所需的最大功率范围很广，任何子状态之间所需的转换时间都是特定于实现的为了使软件能够构建电源管理策略（在本规范的范围之外），该函数定义了两个转换延迟值。每个函数子状态将其最大转换延迟与转换延迟值之一相关联，其中

maximum Transition Latency是函数从任何其他子状态进入配置子状态所需的时间。允许函数以比子状态的最大转换延迟更快的速度完成子状态转换

* + 1. [多功能设备](#_bookmark116)的DPA功能

允许[多功能设备](#_bookmark116)的部分或全部功能实现DPA功能。[多功能设备](#_bookmark116)的功率分配是由DPA能力为每个功能设置的功率分配的总和允许功能的DPA能力包括功能本身的功率分配以及考虑不实现DPA能力的其他功能的功率分配DPA的多个功能之间的关联是特定于实现的，超出了本规范的范围

* 1. TLP处理提示（TPH）

TLP处理提示是一个可选功能，它在请求TLP报头中提供提示，以促进优化处理以内存空间为目标的请求这些处理提示使得系统硬件（例如，根复合体和/或端点）来优化平台资源，例如基于每个TLP的系统和内存互连。TPH机制定义处理提示，其提供关于端点和根复合体之间的通信模型的信息。转向标记是系统特定的值，用于标识编译器明确针对的处理资源系统软件发现并识别TPH功能，以确定支持TPH的每个功能的引导标签分配

* + 1. 处理提示

该查询器向根复合体或其他目标提供关于主机和/或设备对数据和数据结构的预期使用的提示提示由具有即将到来的请求模式的知识的完成者提供，并且完成者将不能够自主地推断（具有良好的需要用这些提示进行区分的感兴趣的情况包括：

DWHR：设备写入，然后主机很快

HWDR：设备读取据信主机最近写入的数据D\*D\*：设备写入/读取，然后设备很快读取/写入

包括DWDW、DWDR、DRDW、DRDR

双向：主机和设备共享并具有相等读/写访问权限的数据结构使用模型映射到处理提示编码，如[表6-12](#_bookmark121)所述。

表6-12处理提示映射

|  |  |  |
| --- | --- | --- |
| PH[1：0]（b） | 处理提示 | 使用模型 |
| 00 | 双向数据结构 | 双向共享数据结构 |
| 01 | 请求者 | D\*D\* |
| 10 | 目标 | DWHR HWDR |

|  |  |  |
| --- | --- | --- |
| PH[1：0]（b） | 处理提示 | 使用模型 |
| 11 | 目标优先级 | 与目标相同，但具有临时重用优先级 |

* + 1. 转向标签

旨在将TLP定向到诸如主机处理器或系统高速缓存分层结构之类的特定处理资源的功能需要目标高速缓存的拓扑信息（例如，哪个主机缓存）。引导标记是系统特定的值，提供有关系统缓存层次结构中的主机或缓存结构的信息这些值用于将平台内的处理元素与请求的处理

将使用的软件可编程转向标签值存储在ST表中，该表允许位于[TPH调压器扩展能力](#_bookmark38)结构中（见[第7.9.13](#_bookmark39)）或与MSI-X表结合（见[第7.7](#_bookmark49)

），但对于给定的函数，不在这两个位置当ST表与MSI-X表结合使用时，每个MSI-X表条目的矢量控制寄存器的2个最高有效字节用于包含引导标签值。

ST表位置的选择取决于具体实现，可通过软件进行验证允许实现MSI-X的函数在任一位置定位ST表（参见[7.9.13.2部分](#_bookmark30)）。允许实现MSI和MSI-X的功能将ST表与MSI-X表组合并使用它，即使在MSI-X被禁用时（即当MSI被启用时）。每个ST表条目为2个字节。ST表的大小在[TPH计数器扩展功能](#_bookmark38)结构中指示。

对于某些使用模型，不需要或不提供转向标签，并且在这种情况下，允许函数在ST字段中使用全零的值来指示没有ST偏好。每个请求与ST表条目的关联是特定于设备的，超出了本规范的范围

* + 1. ST操作模式

[TPH转换器扩展能力](#_bookmark38)结构中的ST表位置字段指示ST表由功能实现的位置（如果有的话）如果实施ST表，软件可以使用系统特定的转向标签值对其进行编程

表6-13 ST操作

|  |  |  |
| --- | --- | --- |
| ST模式选择[2：0]（b） | ST模式名称 | 描述 |
| 000 | 无ST  模式 | 该函数必须对所有转向标记使用全零值 |
| 001 | 矢量模式 | 每个引导标记由MSI/MSI-X中断向量编号选择该函数需要使用ST表条目中的引导标签值，该ST表条目可以通过有效的MSI/MSI-X中断向量编号进行索引。 |
| 010 | 设备特定模式 | 建议功能使用ST表条目中的转向标记值，但这不是必需的。 |
| 所有其他编码 | 保留 | 保留以备将来使用。 |

在无ST操作模式下，该功能必须为每个转向标签使用全零值，以便在没有软件提供的转向标签的情况下使用处理提示

在中断向量操作模式下，使用MSI/MSI-X中断向量编号从ST表中选择引导标记对于已启用MSI的功能，该功能需要在MSI功能结构中的多消息启用字段指定的范围内选择标记对于启用了MSI-X的函数，该函数需要在MSI-X表大小范围内选择标签如果ST表大小小于中断向量数的启用范围，则允许该功能对于某些事务不使用TPH，使用具有为0的引导标签的TPH，或者使用具有用于从ST表中选择引导标签值的实现定义的机制的TPH如果ST表大小大于中断向量编号的启用范围，则功能将忽略与超出范围的中断向量编号

在设备特定操作模式下，将转向标签分配给请求是设备特定的。无论ST表的位置如何，允许功能使用的引导标签的数量不同于为功能分配的中断向量的数量，并且请求中使用的引导标签值不需要来自架构化ST表。

需要能够生成TPH请求的功能来支持无ST操作模式对其他ST操作模式的支持是可选的。通过对ST Mode Select（ST模式选择）进行编程，一次只能选择一种ST操作模式

执行说明

ST扫描床编程

为确保在请求中使用确定性的引导标签值，建议软件在执行ST表更新的过程中暂停功能或禁用TPH触发器功能如果不这样做，可能会导致ST表更新期间使用的ST值的不确定值

* + 1. TPH能力

TPH能力是可选的规范性的。需要能够生成具有TPH的请求TLP的每个功能来实现[TPH验证器扩展能力](#_bookmark38)结构。支持以TPH作为完成器处理TLP的功能需要通过设备功能2寄存器指示TPH完成器功能TPH被架构为应用于以存储器空间为目标的事务，并且适用于设备到主机、设备到设备和主机到设备之间的事务流在要支持TPH的每种情况下，路由器、完成器和所有中间路由元件必须支持相关联的TPH能力。

软件通过[TPH加密器扩展功能](#_bookmark38)结构发现加密器功能，并通过[设备功能2寄存器](#_bookmark13)发现完成器功能（参见[7.5.3.15部分](#_bookmark13)）。软件必须对TPH请求[者扩展能力](#_bookmark38)结构中的TPH请求者启用字段进行编程，以使该功能能够通过TPH发起请求

TPH仅提供附加信息以使得能够对以存储器空间为目标的请求进行优化处理，因此用于管理存储器空间访问的现有机制和规则（例如，总线主使能、存储空间使能和基址寄存器）不改变。

* 1. 延迟容限报告（LTR）机制

延迟容限报告（LTR）机制使端点能够向根复合体报告其存储器读取和写入的服务延迟要求，以便可以实现中央平台资源（例如主存储器、RC内部互连和监听资源）的电源管理策略，LTR机制不直接影响链路电源管理或交换机内部电源管理，尽管可能会产生间接影响

“延迟容限”的含义在不同的设备类型和实现方式之间会有很大差异。当实现该机制时，通常期望考虑服务延迟是否影响功能或仅影响性能，性能影响是否是线性的，以及设备有多少可能使用缓冲和/或其他技术来补偿延迟敏感性。

根复合体不需要遵守所请求的服务延迟，但是强烈鼓励提供不超过由LTR机制指示的延迟的最坏情况服务延迟

LTR支持是通过[第7](#_bookmark161)中描述的报告和控制寄存器发现和启用的。除非根联合体和所有中间交换机指示支持LTR，否则软件不得在端点中启用LTR请注意，并不要求所有端点都支持LTR才允许在支持LTR的端点中启用LTR。在层次结构中启用LTR机制时，必须首先启用最靠近根端口的设备

如果在不支持LTR的下游端口接收到LTR消息，或者LTR未启用，则必须将该消息视为不支持的请求。

无监听/监听延迟[15：0]

15 14 13 12 10 9 0

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

【1：0】

LatencyValue [9：0]

LatencyScale [2：0] -编码：000 -值乘以1 ns

001 -数值乘以32 ns

1. - 数值乘以1，024 ns
2. - 数值乘以32，768 ns
3. - 数值乘以1，048，576 ns
4. - 数值乘以33，554，432 ns 110-111 -不允许

图6-15 LTR消息

粤ICP备17076666号-1

**No-Snoop Latency和Snoop Latency：**如[图6-15](#_bookmark122)所示，这些字段包括一个Requirement bit，指示设备是否对给定类型的请求有延迟要求如果Requirement位为Set，则LatencyValue和LatencyScale字段描述延迟要求。如果Requirement位为Clear，则不存在延迟要求，并且忽略LatencyValue和LatencyScale字段对于任何LTR消息传输，允许设备指示仅针对无监听请求、仅针对监听请求或针对两种类型的请求报告要求还允许设备通过清除两个字段中的Requirement位来指示其对任一类型的流量都没有要求

每个字段还包括对报告的延迟进行编码的值和标度字段将这些值乘以所指示的标度以产生绝对时间值，其可在1 ns至225 \*（210 - 1）= 34，326，183，936 ns的范围内

将值和标度字段设置为全0

如果设备没有实现或没有特定类型流量的服务要求，则必须清除相关延迟字段的Requirement位

当通过写入PMCSR寄存器定向到非D0状态时，如果器件最近发送的LTR消息（自上次DL\_Down到DL\_Up转换以来）报告了一个或两个延迟字段，其中任何要求位均已设置，则在转换到非D0状态之前，必须发送一个新的LTR消息，其中两个要求位均已

清除LTR机制启用位时，如果设备最近发送的LTR消息（自上次DL\_DOWN到DL\_Up转换以来）报告了设置了任何要求位的

* 如果该位因对设备控制2寄存器进行配置写入而被清除，则设备必须发送一条新的LTR消息，其中所有要求位均被清除。
* 如果该位因FLR而被清除，强烈建议设备发送一条新的LTR消息，并清除所有要求位。

当下游端口进入DL\_Down状态时，必须将该端口先前记录的任何延迟视为无效。

来自设备的LTR消息从设备的角度反映了可容忍的延迟，对此，平台必须考虑服务延迟本身，加上使用时钟功率管理（CLKCLK #）增加的延迟（如果适用）。服务延迟本身定义如下：

* 当设备发出非发布请求时，该请求的服务延迟是从发送请求TLP的最后一个符号到接收该请求128的第一个完成TLP的第一个符号的延迟。
* 当设备发出一个或多个张贴请求使得其由于流控制背压而不能发出另一张贴请求时，被阻止请求的服务延迟是从先前张贴请求的最后符号的传输到返回允许被阻止请求118的传输的信用的DLLP的第一符号的接收的延迟。

如果使用时钟电源管理，则在设备断言CLKCLK #和设备接收有效时钟信号之间的平台实现相关时段构成平台服务延迟的附加部分，平台在设置平台电源管理策略时必须理解该

建议端点在启用LTR后立即向上游传输LTR

强烈建议端点在任何500 μs时间段内发送不超过两个LTR消息，除非规范要求如果在500 μs的时间段内接收到两个以上的LTR消息，则下游端口不得生成错误，并且必须正确处理所有LTR消息，而不管它们之间的时间间隔如何

与上游端口相关联的[多功能设备](#_bookmark116)（MFD）必须根据以下规则向上游传输

* 由MFD向上游发送的消息的可接受延迟值必须反映与任何函数相关联的最低
  + 允许在聚集消息中报告的监听和非监听延迟与不同的功能相关联。
  + 如果没有一个功能报告对某种类型的流量（监听/无监听）的要求，则MFD发送的消息不得设置与该类型流量对应的要求位
* 当MFD的任何功能以改变MFD先前报告的聚集值的方式改变其内部报告的值时，MFD必须向上游传输新的LTR消息

1. 对于该定义，包括DLLP或TLP的所有符号对于8b/10 b，第一个和最后一个符号是成帧符号（SDP、STP或END，参见[第4.2.1](#_bookmark6)节）。对于128 b/130 b，数据包的第一个符号是成帧令牌（SDP或STP）的第一个符号，数据包的最后一个符号是CRC或LCRC的最后一个符号（见[第4.2.2](#_bookmark99)）。

交换机必须从启用了LTR机制的下游端口收集消息，并根据以下规则向上游传输

* + 如果交换机支持LTR功能，则它必须在其上游端口和所有下游端口上支持该功能
  + 交换机上游端口仅在其LTR机制启用位被置位时或在软件清除其LTR机制启用位后不久才被允许传输LTR消息，如本节前面所述
  + 交换机向上游发送的消息的可接受延迟值必须计算如下：
    - 如果没有下游端口接收到包含对特定类型流量（监听/非监听）的要求的LTR消息，则交换机发送的任何LTR消息不得设置与该类型流量对应的要求位
    - 将LTRdnport[N]定义为在下游端口N处接收的LTR消息中报告的值，并在适用时进行以下调整：
      * 如果清除了Requirement位或使用了Not Permitted LatencyScale值，则LTRdnport[N]实际上是无限的
      * 如果Requirement位为1且LatencyValue字段全为0，则LTRdnport[N]必须为0，而
    - 将LTRdnportMin定义为跨所有下游端口的LTRdnport[N]的最小值
    - 将Lswitch定义为Switch引起的所有延迟
      * 如果Lswitch基于交换机的操作模式动态地改变，则交换机必须不允许Lswitch超过LTRdnportMin的20%，除非用于交换机的最低延迟模式的Lswitch更大，在这种情况下，
    - 计算要向上游传输的值LTRconglomerated，作为LTRdnportMin-Lswitch，除非该值小于0，在这种情况下，LTRconglomerated为0
    - 如果LTRconglomerated为0，则聚合LTR消息中的LatencyValue和LatencyScale字段必须全为0
  + 如果由于DL\_Down使为该端口记录的先前延迟无效而导致聚集的延迟发生变化，则必须向上游传输新的LTR消息
  + 如果交换机下游端口的LTR机制启用位被清除，则为该端口记录的延迟容限值必须被视为无效，并且要向上游传输的延迟更新，并且如果聚合延迟因此而改变，则向上游传输新的聚合消息
  + 当任何下游端口/功能以改变交换机报告的聚集延迟的方式改变其报告的延迟时，交换机必须向上游传输LTR消息
  + 交换机不得向上游传输LTR消息，除非由上述事件之一触发

允许RC延迟设备请求TLP的处理，只要它满足设备的服务要求。

当延迟要求在一系列请求期间更新时，要求RC在服务后续请求之前理解更新的延迟数字在所有情况下，更新的延迟值必须在等于或小于先前报告的延迟要求的时间段内生效。允许RC比该限制更早地理解更新的延迟数字

一

B

C

D

E

CLK编号

时钟有效器件PLL

pciexpress链路

L1 ASPM

L1 L0 前导DMA事务

粤ICP备17076777号-1

执行说明

LTR的最佳使用

建议端点在每次端点的服务要求发生变化时传输更新的LTR消息如果延迟容限正在减少，则建议在具有新要求的第一个预期请求之前发送更新的LTR消息，从而允许先前发出的LTR消息中指示的时间量如果正在增加容限，则更新应立即跟随具有先前延迟容限值的最终请求

通常，链路将在ASPM L1中，并且如果支持时钟功率管理（时钟PM），则在端点达到使端点向RC发起请求的内部触发时，CLKCLK #将被解除断言以下文本显示了在这种情况下如何应用LTR的示例关键时间点如[图6-16](#_bookmark123)所示。

图6-16 CLKCLK #和时钟电源管理

时间A是器件断言CLKCLK #和器件接收有效时钟信号之间的平台实现相关周期此值不会超过有效的延迟时间

时间B是器件具有有效时钟和可以启动重新训练序列以从L1 ASPM转换到L0之间的器件实现相关周期

时间C是从L1 ASPM到L0的转换发生的时间段

读事务的时间D是请求TLP中的END符号的传输与该请求的完成TLP中的STP符号的接收之间的时间写事务的时间D是从传输耗尽FC信用的TLP的END符号到接收DLLP中返回该请求类型的更多信用的SDP符号此值不会超过有效的延迟时间

时间E是从端点到系统存储器的数据路径打开的时间段，并且数据事务不受前导延迟的影响

LTR延迟语义反映了设备看到的可容忍延迟，通过以下一项或两项测量



A B

A'B

C

D'

E'

CLK编号

时钟有效器件PLL

pciexpress链路

ASPM L1 L0 电极导线LTR

L1 ASPM

L1 L0前导DMA事务

粤ICP备17076888号-1

情况1：设备可能支持也可能不支持时钟PM，但尚未取消断言其CLKCLK #信号-设备观察到的延迟在[图6-16](#_bookmark123)中表示为时间C和D之和。

情况2：设备支持时钟PM并且已经解除断言CLKCLK #-由设备观察到的延迟被表示为时间A、C和D的总和

为了有效地将LTR机制与时钟PM结合使用，设备将知道或能够测量时间B和C，以便它知道何时断言CLKREQ#。时间A、时间C和时间D的实际值可能会动态变化，平台有责任确保总和不会超过延迟。

图6-17使用LTR和时钟电源管理

在一个非常简单的模型中，端点可以选择实现LTR，如[图6-17](#_bookmark124)所示。当端点确定其空闲时，其发送具有软件配置的最大延迟或端点可以支持的最大延迟

当端点确定其需要维持与根复合体的持续数据传输时，端点发送具有较短延迟的新LTR消息（在时间E）。该LTR消息在下一次数据刷新之前发送在时间E和时间

注意，RC可以延迟设备请求TLP的处理，只要它满足设备的服务要求。例如，如果连接到根端口1的端点报告100 μs的延迟容限，并且根端口2上的端点报告30 μs的值，则RC可以实现在以30 μs的延迟服务请求之前在来自根端口1的空闲时段之后将初始请求停止70 μs的策略，从而向第一端点提供100 μs的感知服务延迟这种RC行为为RC提供了将请求批处理在一起以获得更高效服务的能力

在确定RC可以在最大监听和最大无监听时间间隔内为来自所有端点的监听和无监听请求提供服务后，可以通过更新Max Snoop LatencyValue、Max Snoop LatencyScale和Max No-Snoop LatencyValue、Max No-Snoop LatencyScale字段将此信息传递到端点此通信的目的是防止端点发送不必要的LTR更新。

当端点类似地，当端点的无监听请求的LTR值改变为变得比最大无监听请求中指示的值更大（更宽松）时，

LatencyValue/Scale字段，建议端点发送具有Max No-Snoop LatencyValue/Scale字段中指示的无侦听LTR值的LTR消息

建议端点尽可能多地缓冲请求，然后在端点实际支持的突发中使用完整的链路带宽，因为这通常会导致最佳的整体平台功率效率。

注意，LTR可以在并非所有端点都支持LTR的环境中启用，并且在这样的环境中，不支持LTR的端点可能经历次优服务。

* 1. 优化的缓冲区刷新/填充（OBFF）机制

优化的缓冲器刷新/填充（OBFF）机制使得根复合体能够在端点总线主控和/或中断活动的增量平台功率成本相对较低时向端点（贯穿层级）报告时间窗口通常，这将对应于主机CPU、存储器和与根复合体相关联的其他中央资源活动以服务于某个其他活动（例如，操作系统定时器滴答）的时间。这种窗口的性质和确定是平台/实现特定的。

OBFF指示是一个提示-功能仍然被允许启动总线主控和/或中断流量，无论何时启用，尽管这对于平台功率来说不是最佳的，应该尽可能避免

OBFF使用WAKE#信号或消息来指示（参见[2.2.8.9部分](#_bookmark63)）。该消息仅用于WAKE#信号不可用的互连OBFF或CPU活动的WAKE#信令必须仅在系统处于操作状态时由根端口发起，该操作状态在符合ACPI的系统中对应于S0状态。处于非D0状态的功能不能响应OBFF或CPU Active信令。

OBFF消息路由被定义为100b，用于点对点，并且仅允许在下游方向上传输。存在多个区分的OBFF事件当使用OBFF消息时，OBFF代码字段用于区分不同的OBFF情况：

1111 b

所有其他代码均保留。

这些代码对应于使用WAKE#信令时WAKE#的各种断言模式，如[图6-18](#_bookmark125)所示。当发信号通知OBFF时存在一个负向转变，并且每次发信号通知CPU活动时存在两个负向转变使用WAKE#时所需的电气参数在[[CEM-2.0](#_bookmark100)]（或更高版本）的WAKE#信令部分中定义

转换事件

唤醒编号

OBFF消息代码

空闲OBFF

空闲CPU活动

OBFF或CPU活动空闲

OBFF CPU活动

OBFF

CPU活动

空闲

CPU活动

CPU活动OBFF

OBFF

粤ICP备17078666号-1

图6-18代码和等效WAKE#模式

当接收到指示保留代码的OBFF消息时，如果OBFF被启用，则接收器必须将该指示视为

在未实现OBFF的端口或未启用OBFF时收到的OBFF消息必须作为不支持的请求（UR）处理这是与接收端口相关的报告错误（参见[第6.2](#_bookmark18)）。如果一个端口使用WAKE#信令使能了OBFF，并且该端口接收到OBFF消息，则行为未定义。

OBFF指示反映中央资源功率管理状态转换，并且当平台拓扑支持WAKE #时使用WAKE#来发信号，或者当WAKE#不可用时使用消息来发信号OBFF支持是通过[第7](#_bookmark161)中描述的报告和控制寄存器发现和启用的。软件不得在端点中启用OBFF，除非平台支持将OBFF指示传递到该端点。

当平台指示CPU活动或OBFF窗口开始时，建议平台不要在10 μs内返回到空闲状态允许在实际进入平台空闲之前指示返回到空闲状态，但强烈建议仅在防止后期端点活动导致立即退出空闲状态时才这样做，并且提前时间应尽可能短

建议端点不要假设CPU活动或OBFF窗口将在任何特定时间长度内保持打开

WAKE\_1#

根复合体

交换机A

OBFF消息

端点1

交换机B

端点2

终点3

终点4

WAKE\_2#

图6-19显示OBFF由消息

粤ICP备17078777号-1

[图6-19](#_bookmark126)示出了一个示例系统，其中交换机（A）需要将使用WAKE#接收的OBFF指示转换成OBFF消息，在这种情况下，OBFF消息由另一交换机（B）接收并转换回使用WAKE#信令。HwInit配置机制（由硬件或固件设置）用于识别诸如本示例中所示的情况（其中交换机A和交换机B之间的链路需要使用OBFF消息），并且系统固件/软件必须相应地配置OBFF。

当交换机被配置为在其上游端口使用OBFF消息信令并在一个或多个下游端口使用WAKE#时，或者反之亦然，当启用OBFF时，交换机需要将在上游端口接收到的所有OBFF指示转换为下游端口的适当形式

当使用WAKE#时，任何特定根端口的启用将启用WAKE#的全局使用，除非有多个WAKE#信号，在这种情况下，只有关联的WAKE#信号受到影响。当使用OBFF的消息信令时，特定根端口的使能仅使能来自该根端口的OBFF消息的传输为了确保OBFF

如果在平台中完全启用，则必须为OBFF启用指示OBFF支持的所有根端口允许系统固件/软件选择性地使能OBFF，但是这种使能超出了本说明书的范围

为了最小化功耗，强烈建议系统固件/软件仅在WAKE#信令对于给定链路不可用时启用OBFF的消息信令

使用WAKE#的OBFF信令必须仅报告为连接到交换机的所有组件支持，如果它是共享WAKE#信号。在这些拓扑中，允许软件为连接到交换机的组件启用OBFF，即使交换机本身不支持OBFF。

允许（但不鼓励）连续多次指示同一OBFF事件

当交换机正在向下游传播OBFF指示时，强烈建议传播所有OBFF指示。然而，特别是当使用消息时，交换机可能需要丢弃或折叠OBFF指示。当接收到相同或不同类型的指示时，允许丢弃和替换给定类型的较早指示。

下游端口可以被配置为以两种方式传输OBFF消息，这两种方式被称为变型A和变型B。对于变型A，如果链路处于L0状态，则端口必须发送OBFF消息，但是当链路处于Tx\_L0s或[L1](#_bookmark81)状态时丢弃消息。当下游端口通向预期具有非时间关键的通信要求的设备时，并且当预期设备通过将链路状态返回到L0来发信号通知非紧急需要注意时，这种变化是优选的。对于变型B，如果链路处于L0状态，则端口必须发送OBFF消息，或者，如果链路处于Tx\_L0s或[L1](#_bookmark81)状态，则端口必须将链路引导到L0状态，然后发送OBFF消息。当下游端口导致可以从平台状态的及时通知中受益的设备时，这种变化是优选的

当初始配置为OBFF操作时，初始假定指示必须是CPU活动状态，而不管WAKE#信号的逻辑值如何，直到观察到第一次转换

当为OBFF启用端口时，建议在启用下游端口之前启用所有上游端口，并且必须在启用所有其他端口之后启用根端口。对于热插拔端口，通常不可能执行此序列，并且允许使用WAKE#对未连接的热插拔下游端口启用OBFF建议不要为OBFF消息传输启用未连接的热插拔下游端口。

执行说明

OBFF终点注意事项

在正常情况下，可能会合法地发生事件，导致端点误解从空闲窗口到CPU活动窗口或OBFF窗口的转换例如，非OBFF端点可以断言WAKE#作为唤醒机制，屏蔽系统这可能会导致端点的行为方式因电源或性能原因而低于最佳状态，但对于端点或主机系统而言不应不可恢复

为了使端点能够尽可能准确地查看主机状态，建议端点在接收到它确定为主机发起的请求时，以及在端点具有由主机软件服务的挂起中断的任何时间点，将其内部状态跟踪逻辑置于CPU活动状态

* 1. PASID TLP前缀

PASID TLP前缀是[第2.2.1](#_bookmark50)中定义的端-端TLP前缀。PASID TLP前缀的布局如所示

[图6-20](#_bookmark128)和[表6-14](#_bookmark129)。

当存在PASID TLP前缀时，前缀中的PASID值与请求者ID一起标识与请求关联的进程地址空间ID每个函数都有一组不同的PASID值。一个函数使用的PASID值与任何其他函数使用的PASID值无关

允许PASID TLP前缀用于：

* 内存请求（包括AtomicOp请求）与未翻译的错误（参见[2.2.4.1部分](#_bookmark79)）。
* 地址转换请求、ATS无效消息、页面请求消息和PRG响应消息（参见[第10.1.3节](#_bookmark64)）。

在任何其他TLP上都不允许使用PASID TLP前缀

* + 1. 管理PASID TLP前缀使用

必须专门启用PASID TLP前缀的使用除非启用，否则不允许组件传输PASID TLP前缀。

对于端点功能（包括根综合体集成设备），以下规则适用：

* + - * 不允许函数发送和接收带有PASID TLP前缀的TLP，除非设置了PASID启用（请参见

[部分7.8.8.3](#_bookmark65)）：

* + - * 一个函数必须有一个机制，用于动态地将PASID的使用与一个特定的函数上下文相关联。该机制是特定于器械的
      * 函数必须有一种机制来请求它正常停止使用特定的PASID。此机制是设备特定的，但必须满足以下规则：
        + 一个函数可以支持有限数量的同时PASID停止请求。软件应该推迟发出新的停止请求，直到旧的停止请求完成。
        + 一个函数中的停止请求不得影响任何其他函数的操作
        + 停止请求不得影响函数内任何其他PASID的操作
        + 停止请求不得影响与PASID无关的事务的操作
        + 当停止请求机制指示完成时，该功能具有：

已停止对该PASID的新请求进行排队

已完成与此PASID关联的所有未发布请求

将PASID使用的所有TC中寻址主机内存的所有Posted Request刷新到主机用于此的机制是特定于设备的（例如：对主机存储器的非松弛Posted Write或对函数的处理器读取可以刷新TC 0;对主机存储器的零长度读取可以刷新非零TC）。

（可选）将所有对等发布请求刷新到其用于此的机制是特定于设备的

如果代表本PASID发布地址翻译或页面请求，则遵守地址翻译服务（[第10](#_bookmark4)）中描述的其他规则

对于根复合体，以下规则适用：

* + - * 根复合体必须具有特定于设备的机制，用于指示对PASID TLP前缀的支持
      * 支持PASID TLP前缀的根复合体必须具有用于启用它们的设备特定机制默认情况下，禁用PASID TLP前缀。
      * 支持PASID TLP前缀的根复合体可以可选地具有用于以比整个根复合体更精细的粒度来启用它们的设备特定机制（例如，用于特定根端口、路由器ID、总线号、路由器ID或路由器ID/PASID组合的不同启用
    1. PASID TLP布局

一个TLP最多可以包含一个PASID TLP前缀。

执行所请求

+0

7 6 5 4 3 2 1 0

+1

+2

+3

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

100 1 0001 R PASID

已请求的挂起模式

图6-20 PASID TLP前缀

表6-14 PASID TLP前缀

|  |  |
| --- | --- |
| 比特 | 描述 |
| 字节0  比特7：5 | 100 b-指示TLP前缀 |
| 字节0位4 | 1b-表示端-端TLP前缀 |
| 字节0  比特3：0 | 0001 b-表示PASID TLP前缀 |
| 字节1位7 | ***已请求的重命名模式***- 如果设置指示端点中的重命名模式实体正在发出请求，如果清除，则指示端点中的非重命名模式实体正在发出请求。  此位的用法在[6.20.2.3节](#_bookmark133)中有详细说明。 |
| 字节1位6 | ***Execute Requested***- 如果设置指示端点正在请求执行权限。如果为Clear，则指示终结点未请求执行权限。  此位的用法在[6.20.2.2节](#_bookmark132)中有详细说明。 |
| 字节1位3：  字节3位0 | **进程地址空间ID（PASID）**- 此字段包含与TLP关联的PASID值此字段的用法在[6.20.2.1一节](#_bookmark131)中定义。 |

* + - 1. PASID字段

PASID字段标识与请求相关联的用户进程此字段存在于所有PASID TLP前缀中。

PASID字段为20位宽。端点和根复合体不需要支持字段的整个范围对于终点，最大PASID宽度字段指示PASID值的支持范围（第7.28.2节）。对于根复合体，使用实现特定的机制来提供该信息。

除非设置了PASID启用位（第7.28.3节），否则不允许端点发送带有PASID TLP前缀的TLP支持PASID TLP前缀的端点在收到带有PASID TLP前缀的TLP且PASID启用位为清除时，必须发出不支持请求（UR）的

根复合体可以选择支持带有PASID TLP前缀的TLP用于检测根复合体是否支持PASID TLP前缀的机制是特定于实现

对于端点，以下规则适用：

* + - * + 不允许端点发送PASID值大于或等于2Max PASID Width的TLP。
        + 当端点接收到PASID值大于或等于2最大PASID宽度的请求时，它可以选择性地发出错误信号。这是与接收端口关联的不支持请求错误（请参见

[第6.2节](#_bookmark18)）。

对于根复合体，以下规则适用：

* + - * + 不允许根联合体发送PASID值大于其支持值的TLP
        + 可选地，当根复合体接收到具有大于其支持的PASID值的请求时，允许根复合体发出错误这是一个与接收端口相关的不支持请求错误（参见[第6.2](#_bookmark18)）。

对于完成者，以下规则适用：

* + - * + 对于未转换的存储器请求，PASID值和未转换的地址都用于确定满足请求所使用的转换地址

对于与地址转换相关的TLP，此字段的用法在地址转换服务（[第10](#_bookmark4)）中定义

执行说明

PASID宽度均匀性

每个函数的PASID值是唯一的，因此最初的意图是该函数支持的PASID值的宽度可以基于该函数的需要。然而，当前的系统软件通常不遵循该模型，而是在访问特定地址空间的所有函数中使用相同的PASID值为了实现这一点，系统软件通常将确保用于根复合体和持久转换代理的公共系统PASID宽度。这样的系统软件通常将在任何热插拔端点功能或报告PASID宽度支持小于公共系统PASID宽度的转换代理上禁用ATS

根复合体、端点和转换代理通常独立于系统软件实现，因此强烈建议硬件实现20位的最大宽度，以确保与系统软件的互操作性。

端点可以以特定于实现的方式能够将20位系统PASID映射到携带较小宽度的内部表示。如果这样做，端点这样做而不影响系统软件是至关重要的，因为系统软件没有机制来区分这种实现与那些本机实现完整20位宽的实现

* + - 1. 执行所请求

如果Execute Requested位被设置，则端点正在请求允许端点执行与此请求相关联的内存范围中的指令Execute权限的含义不在本规范的范围内。

除非执行许可支持位（[第7.8.8.2](#_bookmark101)）和执行许可启用位（[第7.8.8.3](#_bookmark65)）均已设置，否则端点不允许发送设置了执行请求位的TLP

对于根复合体，以下规则适用：

* + - * + 对根复合体请求的执行的支持是可选的。用于确定根复合体是否支持请求执行的机制是特定于实现的
        + 支持执行请求位的根复合体应该具有特定于实现的机制以使其能够使用该位。
        + 支持执行请求位的根复合体可以具有实现特定的机制以使得能够以更精细的粒度（例如，用于特定的根端口、特定的总线号、特定的路由器ID或特定的路由器ID/PASID组合），其默认值是特定于实现的。

对于完成者，以下规则适用：

* + - * + 完成者有位的有效值的概念对于给定的请求，如果执行请求位被支持并且它的使用被启用用于请求，则该位的有效值是请求中的值;否则该位的有效值是0 b。
        + 对于未转换的内存读取请求，完成器使用位的有效值作为保护检查的一部分如果此保护检查失败，则完成程序将请求视为未映射内存
        + 对于未转换内存请求，除未转换内存读取请求外，该位为保留位。对于与地址转换相关的TLP，此位的用法在地址转换服务（[第10](#_bookmark4)）中定义
      1. [已请求的挂起模式](#_bookmark130)

如果设置了“[请求的重命名模式](#_bookmark130)”，则端点将发出一个以与重命名模式相关联的内存为目标的请求如果已[请求的重命名模式](#_bookmark130)为清除，则端点正在发出以与非重命名模式相关联的内存为目标的请求

保护模式和非保护模式的含义以及端点在保护模式或非保护模式下运行的含义取决于系统的保护模式，不在本规范的范围

端点不允许发送已设置“[请求重命名模式](#_bookmark130)[”](#_bookmark51)位的TLP，除非已设置“支持[重命名模式](#_bookmark51)”位（[第7.8.8.2](#_bookmark101)）和“启用[重命名模式”](#_bookmark40)位（[第7.8.8.3](#_bookmark65)）

对于根复合体，以下规则适用：

* + - * + 根复合体对[重命名模式请求](#_bookmark130)位的支持是可选的。用于确定根复合体是否支持[重命名模式请求](#_bookmark130)位的机制是特定于实现
        + 一个根复合体，支持被[请求的](#_bookmark130)模式位应该有一个实现特定的机制，使它能够使用该位。
        + 支持[特权模式请求](#_bookmark130)位的根复合体可以具有特定于实现的机制，以使得能够以更细的粒度使用该位（例如，对于特定的根端口、对于特定的总线号、对于特定的请求者ID或对于特定的请求者ID/PASID组合）。

对于完成者，以下规则适用：

* + - * + 完成者有位的有效值的概念。对于给定的请求，如果支持[重命名模式请求](#_bookmark130)位并且为请求启用其使用，则该位的有效值为请求中的值;否则该位的有效值为0 b。
        + 对于未转换的内存请求，完成器使用位的有效值作为其保护检查的一部分如果此保护检查失败，则完成程序将请求视为未映射内存
        + 对于与地址转换相关的TLP，此位的用法在地址转换服务（[第10](#_bookmark4)）中定义
  1. 轻量级通知（LN）协议

轻量级通知（LN）协议使端点能够在主机存储器中的特定高速缓存行中注册兴趣，并且当它们被更新时经由硬件机制被通知。LN转发器（LNR）是端点中的客户端子系统，其发送LN读/写请求并接收LN消息。LN完成器（LNC）是主机中的服务子系统，其接收LN读/写请求，并且在更新注册的高速缓存行时发送LN消息

LN协议为感兴趣的缓存行更新时提供通知服务最常见的是，LNR向具有关联LNC的内存空间范围发送LN Read，请求缓存行（本节中缩写为“行”）的副本LNC将所请求的行返回给LNR，并记录LNR已在该行更新时请求通知;也就是说，LNC稍后，当某个实体更新线路时，LNC通过LN消息通知LNR，因此LNR可以采取适当的动作。对于LN写入存在类似的通知服务，其中写入行的LNR可以在稍后更新该行时被通知

LN协议允许多个LNR同时注册同一行。允许具有LNR的端点随时写入行，无论LNR是否已注册该行。

典型的系统由主机处理器、主机内存、主机内部结构、根端口、交换机和端点组成

下面的[图6-21](#_bookmark134)说明了元素的简化视图以及它们如何互连以提供用于描述LN协议如何操作的上下文

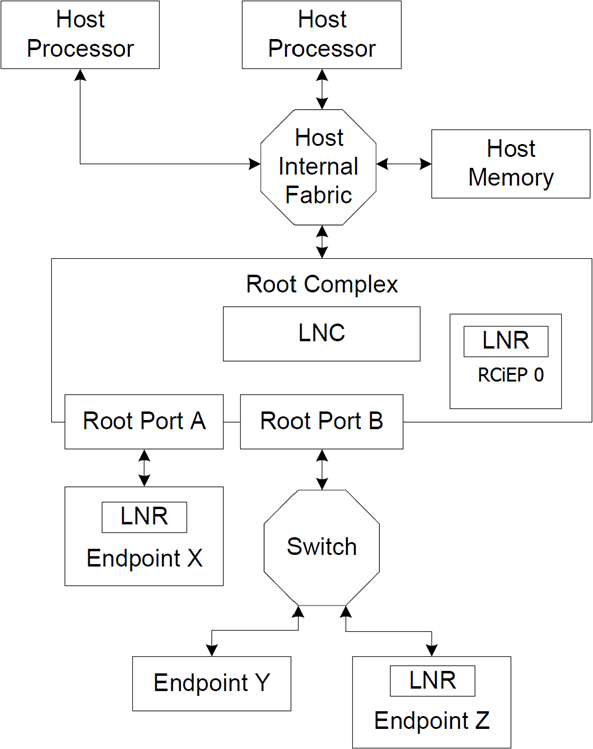


图6-21 LN系统框图示例

在上图中，端点X、端点Z和RCIEP 0均包含一个LNR。根复合体包含LNC。

* + 1. LN协议操作

LN是一个简单的协议，满足几个关键使用模型，以最小的复杂性和成本。

LN过滤器（LNR）

LN完成者（LNC）

LN过滤器（LNR）

LN完成者（LNC）

*图6-22 LN协议基本操作*

具有单个LNR的LN协议操作如上图所示对于读取的情况，如左图所示：（1）LNR使用LN Read从主机存储器请求行的副本;（2）LNC在LN Completion中返回该行，并记录LNR已注册该行;以及（3）LNC稍后在已更新注册行时使用LN消息通知LNR。对于写入的情况，如右侧所示：（1）LNR使用LN Write将行写入主机存储器;（2）LNC记录LNR已注册该行;以及（3）LNC稍后在已更新注册行时使用LN消息通知

LNC必须发送LN消息，或者如果注册线路被某个实体更新（例如，CPU或设备），或者如果LNC不再跟踪该线路的状态后一种情况被称为驱逐，并且由LN消息中的通知原因（NR）字段指示

如果LN转发器对其已经注册的行进行LN读或LN写，则LN转发器通常不能确定其针对该行接收的后续LN消息是针对最近的LN读/写请求还是针对先前的LN读/写

LN协议允许多个LNR同时注册同一行在这种情况下，LNC通过向每个LNR发送定向LN消息或者通过向每个受影响的层次域发送广播LN消息来通知多个

一旦LNC更新或驱逐给定线路并发送一个或多个LN消息以通知具有相关联的注册的所有LNR，则LNC将不发送针对该特定线路的附加LN消息，除非其接收到对该线路的新LN读取或

允许LNC对每个给定行可以独立跟踪的LNR数量有特定的实现限制。在此限制或低于此限制时，LNC通常使用定向LN消息进行通知。超过此限制，LNC使用广播LN消息。一个实现被允许有一个零的限制，并且只使用广播LN

具有特定NR字段值的单个LN消息可以指示注册到该LNR的所有线路已经被驱逐。这种“全部逐出”LN消息的定向和广播版本都

允许LNC对它可以同时注册多少行或行集有特定的实现限制。如果LNC接收到注册新线路的请求，但是LNC没有足够的资源来这样做，则LNC

被允许驱逐一个或多个旧线路以便释放必要的资源，或者发送指示新线路被驱逐的LN消息

执行说明

过度使用广播LN消息

为了避免性能问题，应当实现使用广播LN消息的LNC以最小化每个广播LN消息被发送到的层次域的数量，并且还将广播LN消息的速率保持在合理的界限内。每个广播LN消息消耗链路带宽，并且一些端点可以以相对低的速率处理广播LN消息

具体地，不支持LN转发器能力的端点可以使用低性能机制（例如，用设备固件而不是硬件解码消息。每个消息可以想象地花费微秒来处理129，并且它们的过度速率

虽然不支持LN转发器能力的端点也可以使用低性能机制来处理定向LN消息作为例外情况，但这不应导致性能问题。利用LN协议，LNC仅将定向LN消息发送到支持LN转发器能力的端点，并且假定那些端点将能够以避免性能问题的速率处理定向和广播LN消息两者

* + 1. LN注册管理

由于LNC具有用于注册高速缓存行的有限资源，因此每个LNR中的LNR能力结构为软件提供了限制LNR的未完成注册的最大数量的机制。软件读取LNR配准最大值字段，以发现LNR能够具有未完成的最大数量如有必要，软件可设置LNR配准限制字段以施加指定限制。

LNC注册资源限制可以是高度特定于实现的，可能包括诸如集合关联性、集合的最大数量、以及用于主机存储器的不同区域的不同资源集合之类的因素软件如何发现这些资源限制超出了本规范的范围

为了管理其未完成注册的数量，LNR可以通过向它想要注销的每行发送零长度LN Write来注销未完成注册

LNC不需要接受主机内存中所有位置的注册然而，接受注册的存储器区域的粒度要求不比对齐的4KB区域更细为了确定给定的对齐的4KB区域是否如果LNR希望在

* + 1. LN订购注意事项

LN读取具有与其他存储器读取请求相同的排序要求，并且LN写入具有与其他存储器写入请求相同的排序要求LN完成（用于LN读取）与其他内存读取的完成具有相同的排序要求LN消息具有与其他已发布请求相同的排序要求

1. Posted Request Acceptance Limit允许端点最多花费10 μs来处理每个收到的Posted Request。参见[第2.3.1](#_bookmark95)。

对于给定的行，当LN完成器接收到LN读取，随后是对该行的更新时，LN完成器被允许以任一顺序发送LN完成和LN消息

对于给定行，当LN完成器接收到触发驱逐LN消息的LN读取时，LN完成器被允许以任一顺序发送LN完成和LN消息

对于给定的行，当LN完成器接收到触发驱逐LN消息的LN写，随后是对该行的更新时，LN完成器被允许以任一顺序发送两个LN消息

对于不同的行，允许LN完成者以任何顺序发送LN消息

* + 1. LN软件配置

LN协议支持2种高速缓存行大小（CLS）-64字节和128字节。对每个CLS的支持是可选的，无论是根复合体还是端点。对系统有效的CLS由主机确定，并由适用根端口和[RCRB](#_bookmark169)中的LN系统CLS字段指示。指示LN完成程序支持的所有根端口和[RCRB](#_bookmark169)必须指示相同的CLS;否则，结果是未定义的。在任何操作系统运行时，主机都不能更改系统CLS;否则，结果是未定义的。

支持LN协议的端点必须支持一个或两个CLS，并通过LNR-64支持和

LNR-128支持的能力位。使能每个LN触发器时，软件必须确保相关的LNR CLS控制位配置为与系统CLS匹配;否则，结果不确定。仅支持一个CLS的LN译码器被允许将其LNR CLS控制位硬连线到相应的值。

软件不得更改LNR CLS控制位或LNR配准限制字段的值，除非其LNR启用控制位已被清除或正被同一配置写入清除;否则，结果未定义。如果LNR使能位已被清除，则允许软件在设置LNR使能位的同时，使用单个配置写入来更改LNR CLS位和LNR注册限制字段的值

允许软件在任何时候清除LNR使能位当LNR使能位为清除时，LNR必须清除其所有内部注册状态。

* + 1. LN方案总结

LN协议的详细规则和要求分布在本规范的其余部分，但这里是一个一般性的总结加上一些独特的要求。

* + - * LN读取是其TLP报头具有LN位置位的存储器读取请求LN完成是其TLP报头具有LN比特设置的LN写是其TLP报头具有LN位置位的存储器写请求
      * 除非另有明确说明，否则内存读取请求的所有要求均适用于LN读取
        + LN读取必须访问不超过一个高速缓存行，由系统CLS确定LN完成程序必须将违反此规则的情况作为完成程序中止处理，除非完成程序检测到更高优先级的错误。允许部分缓存行LN读取，包括零长度LN读取
        + 如果LN完成程序将LN读取作为不可纠正错误或建议性非致命错误处理，则LN完成程序不得为该请求注册通知服务
        + LN完成器必须将零长度的LN读取处理为目标存储器区域的“探测”，参见[第6.20.2](#_bookmark127)。如果完成状态是成功完成，则TLP报头中的LN位必须指示该区域是否支持注册能力，但是LNC不得为这种情况创建注册。LN完成者必须支持粒度不超过对齐的4KB区域的注册功能。
        + LN读取的排序和流控制规则与存储器读取请求的排序和流控制规则相同
      * 除非另有明确说明，否则内存读取完成的所有要求均适用于LN完成
        + LN读取的完成必须是LN完成（即，在其TLP报头中设置LN位），如果完成者是LN完成者，则目标存储器区域接受注册，并且完成状态是成功完成;否则完成必须在其TLP报头中设置LN位清除请注意，中毒的完成将具有成功完成的完成状态。请参见[2.7.2.2一节](#_bookmark0)。
        + 用于LN完成的排序和流控制规则与其TLP报头具有LN位清除的完成相同
      * 除非另有明确说明，否则内存写入请求的所有要求均适用于LN写入。
        + 根据系统CLS的决定，LN Write只能访问单个高速缓存行LN完成程序必须将违反此规则的情况作为完成程序中止处理，除非完成程序检测到更高优先级的错误。
        + 如果LN完成程序将LN写入作为不可纠正的错误处理，则LN完成程序不得为该请求注册通知服务。请注意，根据其配置，完成程序可能会将中毒LN写入作为不可纠正错误、建议性非致命错误或屏蔽错误进行处理
        + LN完成程序必须将零长度的LN写入作为注销现有注册的请求进行参见[第6.20.2](#_bookmark127)。如果零长度LN写入所针对的高速缓存行之前未注册，则它必须保持未注册状态。
        + LN写入的排序、流控制和数据中毒规则与存储器写入请求的规则相同
        + 中断器不得为MSI或MSI-X中断生成LN写入。LN完成程序必须将以中断地址范围为目标的LN写入作为完成程序中止处理，除非完成程序检测到更高优先级的错误。
      * 对于LN读和LN写，地址必须是正确的类型，如地址类型（AT）字段所示参见[2.2.4.1部分](#_bookmark79)。正确的类型取决于是否使用了翻译代理（TA）看到

[第10章](#_bookmark4)

* + - * + 如果正在使用TA，则地址必须是转换后的地址。为了获取和使用转换后的地址，LN转换器必须支持ATS
        + 如果未使用TA，则地址必须为默认/未转换地址。
        + LN完成程序检测到违反上述规则时，必须将请求作为完成程序中止（CA）处理，除非检测到更高优先级的错误
  1. 精密时间测量（PTM）机制
     1. 介绍

精确时间测量（PTM）通过独立的本地时钟实现跨多个组件的事件精确协调。通常，由于各个时钟对时间的值和变化率有不同的概念，这种精确的协调将是困难的为了解决此限制，PTM使组件能够计算其本地时间与共享PTM主时间（与PTM根关联的独立时域）之间的关系

增强型精确时间管理（ePTM）对PTM设备提出了额外的要求支持ePTM由ePTM Capable位指示

PTM定义如下：

* + - * PTM转换器-能够将PTM用作与端点或上游端口相关联的消费者的功能
      * PTM响应器-能够使用PTM提供与端口或端口相关联的PTM主时间的功能。

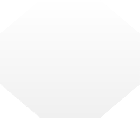
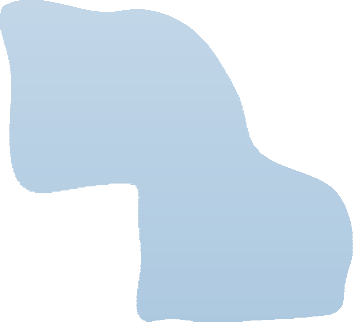
[RCRB](#_bookmark169)。

* + - * 时间源-与PTM响应器关联的本地时钟
      * PTM根-PTM层次结构的PTM主时间的来源PTM根也必须是时间源，并且通常也是PTM响应者。

每个PTM根向所有PTM层次结构提供单个PTM主时间：与单个PTM根相关联的一组PTM计时器

[图6-23](#_bookmark135)显示了使用PTM的一些示例系统拓扑这些仅是说明性示例，并不旨在暗示任何限制或要求。

根复合体是具有单个PTM时间源的PTM根，用于这些PTM层次和根复合体集成端点（RCIEP）。



带PTM的交换机

此交换机是此

PTM

层次结构

此交换机未实现时间源

带PTM的交换机

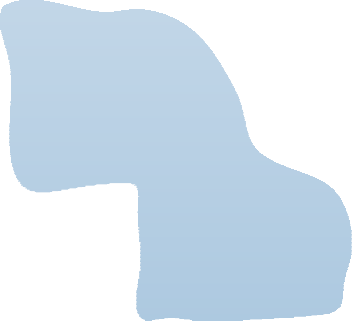
终点（含PTM）

终点（含PTM）

端点

终点（含PTM）

端点



终点（含PTM）

端点

根复合体RCiE

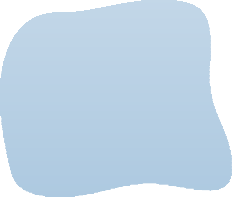
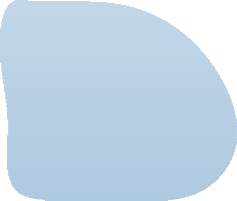
带PTMP

带PTM的交换机

终点（含PTM）

终点（含PTM）

端点



关键词：

每个交换机都是其PTM层次结构的PTM

两个PTM层次具有独立的PTM主时间

PTM应答器

PTM筛选器+响应器PTM根

切换 开关

带PTM 带PTM

终点终点

带PTM带PTM

终点（含PTM）

终点（含PTM）

端点

端点

根复合体

图6-23使用PTM

执行说明

PTM和重定时器

PCIe重定时器会引入不对称链路延迟，从而影响PTM准确性。设计用于保持对称链路延迟的重定时器将实现最佳PTM精度。非对称性越大、越可变，对PTM的影响越大。请参阅制造商的文档，以确定重定时器实现与PTM一起使用的适用性。

* + 1. PTM链路协议

当在链路上的两个组件之间使用PTM时，代表PTM应答器的上游端口将PTM请求发送到同一链路上代表PTM应答器的下游端口。

下游端口

t2 t3

T2'

T3'

t2“

t3“

第2个 PTM对话框

PTM

响应

PTM

响应D（t2 '，t3

第3个 PTM对话框

PTM

响应D（

第1个PTM

对话

PTM

请求

PTM

请求

PTM

请求

t1 t4

**t1't4'**

上游端口

**t1'’**

图6-24精密时间测量链路协议

[图6-24](#_bookmark136)说明了PTM链路协议。上图中的点t1、t2、t3和t4表示每个端口在发送和接收PTM消息时本地捕获的与每个端口相关联的组件将来自第一个PTM对话的这些时间戳存储在内部寄存器中，以供第二个PTM对话使用，并以此类推用于后续PTM对话。

上游端口代表PTM调度器通过发送PTM请求消息来启动PTM对话

下游端口代表PTM应答器，具有对PTM主时间的知识或访问（直接或间接）。

在每个对话期间，下游端口根据先前PTM对话期间存储的时间戳填充PTM ResponseD消息，如[第6.22.3.2](#_bookmark139)中所定义。

一旦每个组件具有来自上述对话框的历史时间戳，与上游端口关联的组件就可以将其时间戳与PTM ResponseD消息中传递的时间戳结合起来，以使用以下公式计算PTM主时间

*t*1′时的PMT主时间

（（*t*4 −*t* 1）−（*t*3 −*t* 2））

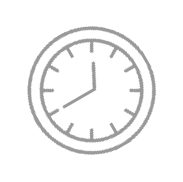
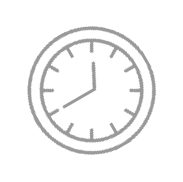
=2

公式6-2 PTM主时间

值t1、t2、t3、t4和t2

具有PTM能力的组件通常将记录这些时间戳计算的结果，并且可以经由实现特定的手段使在本文中，本文档将该结果定时信息称为组件

对于实现PTM的交换机，交换机本身内的时间同步机制是特定于实现的。



在此示例中，根端口提供PTM主机时间。

根复合体

每个链路上的PTM对话交换使组件能够测量其本地时钟与PTM主时间之间的关系。

交换机使用特定于实现的方式将时间从其上游端口传送到其下游端口。

开关

端点

PTM的一个可能用途是使软件能够用独立时钟同步组件之间的事件。

执行说明

PTM理论和操作

在PTM对话框期间捕获的时间戳可以计算PTM应答器和PTM响应器之间的时序关系值（t3-t2）测量PTM响应器针对给定PTM对话所消耗的时间时间（t4-t1）是从请求到响应的时间因此，（（t4-t1）-（t3-t2））有效地给出了两个组件之间的往返消息传输时间，并且该量除以2近似于链路延迟-t1和t2之间的时间差假设从PTM请求者到PTM响应者以及返回的链路传输时间是对称的，这通常是一个很好的假设（另请参阅关于PTM时间戳捕获机制的实施说明

端点

图6-25精确时间测量示例

[图6-25](#_bookmark137)显示了一个简单的采用PTM的设备层次结构每个上游端口启动PTM对话框，以建立其本地时间与根端口提供的PTM主时间之间的关系

在本例中，交换机在其上游端口上发起PTM对话，以获取PTM主时间，用于实现在其下游端口处接收到的PTM请求消息此交换机采用特定于实现的方式将PTM主时间从其上游端口传送到其下游端口。

支持PTM的组件可以使其PTM上下文可用于软件检查，从而使软件能够在本地时间和PTM主时间之间转换定时信息反过来，这种能力使软件能够以非常高的精度协调多个组件之间的事件

同样，强烈建议实施PTM的平台也将PTM主时间提供给软件。

* + 1. 配置和操作要求

软件不得在与上游端口相关的功能上的PTM控制寄存器中设置PTM使能位，除非链路上的相关下游端口在其相关PTM控制寄存器中已经设置了PTM使能位

功能对PTM的支持通过PTM扩展能力结构的存在来指示不要求层次结构中的所有端点都支持PTM，也不要求软件在支持PTM的所有端点中启用PTM

如果PTM消息由不支持PTM的端口接收，或在PTM启用位清零时由下游端口接收，则该消息必须被视为不支持的请求。这是与接收端口相关的报告错误（参见[第6.2](#_bookmark18)）。由支持PTM的上游端口接收的正确形成的PTM响应，但其PTM启用位被清除，必须被静默丢弃。

通过PTM观察，PTM主时间必须满足以下行为要求：

* + - * 时间值必须是单调的，并且严格递增。
      * 感知的粒度不得大于PTM能力寄存器的本地时钟粒度字段中报告的值
      * 感知时间的开始时间不得晚于PTM根处理其第一个PTM请求消息的时间。

参考[图6-24](#_bookmark136)，以下规则定义了时间戳捕获：

* + - * 当传输PTM请求消息时，即使该传输是重放，PTM验证器也必须更新其存储的t1时间戳
      * 当接收到PTM请求消息时，PTM响应方必须更新其存储的t2时间戳，即使接收到的TLP是重复的。
      * 当传输PTM响应或ResponseD消息时，PTM响应器必须更新其存储的t3时间戳，即使该传输是重放。
      * 当接收到PTM响应消息时，即使接收到的TLP是重复的，PTM验证器也必须更新其存储的t4时间戳
        + 时间戳必须基于构成TLP的STP符号或令牌，就像在端口引脚上观察该符号或令牌的第一位一样通常情况下，这需要进行特定的调整，以补偿无法直接测量实际引脚上的时间，因为时间通常是在Rx或Tx路径中的某个内部点测量的。此测量的精度和一致性不受本规范的限制，但强烈建议实现最高的实际精度和一致性
      1. PTM调试员角色
         * 通过设置PTM能力寄存器中的PTM调试器能力位来指示对PTM调试器角色的支持
         * 仅当启用PTM时，才允许PTM调试器请求PTM主时间指导PTM调度器发出此类请求的机制是特定于实现

上游端口通过PTM对话框获取PTM主时间，如[第2.2.8.10](#_bookmark52)所述。

RCiEP请求PTM主时间的机制是特定于实现

* + - * + 一旦发出PTM请求消息，上游端口在收到PTM响应消息、PTM响应消息、复位之前，或在100 μs内没有来自下游端口的相应PTM消息之前，
        + 收到PTM响应后，上游端口在发出另一个PTM请求消息之前必须等待至少1 μs
        + 对于包含多个PTM转换器的[多功能设备](#_bookmark116)（MFD），与该MFD关联的上游端口必须在每次PTM上下文刷新期间发出单个PTM对话框MFD中的PTM调试器使用此设备范围PTM对话框维护其各自的PTM上下文从一个PTM对话框刷新多个PTM上下文的机制是特定于实现的
        + 强烈建议上游端口在发生以下任何情况时使其内部PTM上下文如果支持ePTM，则当发生以下任何情况时，上游端口必须使其内部PTM上下文

PTM请求被重放。

收到重复的PTM ResponseD TLP

PTM主时间与上游端口的本地时间之间的关系会发生例如，这可能由于转换到非D0状态或由于累积的PPM漂移而发生

这些事件在图6-26中的“本地时间无效事件”标签下分组

* + - * + 如果支持ePTM，则上游端口在重放PTM TLP时必须使其PTM上下文无效，直到两个连续PTM对话成功完成且没有重放。

开始

转换为PTM启用

PTM上下文无效

本地时间无效事件

触发事件

收到PTM

等待>=1µs

发布PTM请求

收到PTM

有效PTM上下文

触发事件

图6-26 PTM调压器操作

执行说明

收到重复TLP时PTM无效

重复的TLP在数据链路层中被检测并丢弃，而PTM消息在事务层中被识别在一些实施方式中，将重复的PTM TLP与其他重复的TLP区分开可能是困难的或过于复杂的。

由于允许上游端口根据特定于实现的标准使其内部PTM上下文无效，因此允许PTM验证器在接收到除任何重复PTM TLP之外的任何重复TLP时使其内部PTM上下文类似地，如果支持ePTM，则允许PTM响应方在接收到任何重复TLP时使其历史时间戳（t2-t3）

* + - 1. PTM响应者角色
         * 通过设置PTM能力寄存器中的PTM响应器能力位来指示对PTM响应器角色的支持
         * 交换机和根联合体被允许实现PTM响应者角色。

当通过设置与交换机上游端口相关的PTM控制寄存器中的PTM启用位来启用PTM时，具有PTM功能的交换机必须响应在其任何下游端口接收到的所有PTM请求消息。

根复合体向RCiEP传送PTM主时间的机制是特定于实现的。

* + - * + PTM响应者必须按如下所示填写PTM ResponseD消息（参见[图6-24](#_bookmark136)和随附的实施说明）：

PTM主时间字段是一个64位值，包含接收到当前PTM对话的PTM请求消息时的PTM主时间值在[图6-24](#_bookmark136)中，对于第二个PTM对话框，这是时间t2 '处的PTM主时间

传播延迟字段是一个32位值，包含接收PTM请求消息和发送先前PTM对话的PTM响应消息之间的间隔在

[图6-24](#_bookmark136)，对于第二个PTM对话框，这是在第一个PTM对话框期间捕获的t2和t3之间的时间间隔

两个场的测量单位均为1 ns。

具有多个下游端口的PTM应答器必须使用来自其所有PTM端口下游端口的单个PTM根的

* + - * + 根据以下规则，充当PTM响应方的交换机下游端口和根端口必须使用PTM响应或PTM响应D来响应在其下游端口处接收到的每个PTM请求消息

PTM响应方在未首先接收到PTM请求消息的情况下不得发送PTM响应或PTM响应D消息

收到PTM请求消息后，PTM响应方必须在10 μs内尝试发出PTM响应或PTM ResponseD消息。

当下游端口没有有效的历史时间戳（t3-t2）来实现PTM请求消息时，PTM响应器必须发出PTM响应。

如果支持ePTM， PTM响应者必须在重放任何PTM响应或PTM ResponseD时立即使其历史时间戳（t3-t2）PTM响应方在收到任何重复的PTM请求后，必须使其历史时间戳（t3-t2）

* + - * + 当PTM应答器已经存储了填充PTM应答D消息所需的值的副本时，PTM应答器必须发出PTM应答D：历史时间戳（t3-t2）和在接收到最近的PTM请求消息时的PTM主时间（时间
        + 当PTM响应器已经存储了历史时间戳（t3）的副本时，允许PTM响应器发出PTM响应

- t2），但必须从别处请求PTM主时间。在这种情况下，如果预计检索时间超过10 μs，则允许在检索PTM主时间时发出PTM响应消息以响应PTM请求消息。

* + - * + PTM响应器传输的历史时间戳和PTM主时间值的感知粒度不得超过PTM能力寄存器的本地时钟粒度字段中报告的粒度
      1. PTM时间源角色-特定于交换机的规则

除了上面列出的PTM应答者和PTM响应者角色要求外，交换机还必须遵循以下要求：

* + - * + 当上游端口与[多功能设备](#_bookmark116)相关联时，仅允许与该上游端口相关联的单个功能对于交换机，与交换机相关的所有PTM功能必须通过该结构进行控制不要求实现PTM扩展能力结构的功能是交换机上游端口功能。
        + 交换机的PTM扩展功能结构必须指明对PTM应答器和PTM响应器角色的支持
        + 上游端口中的PTM扩展功能控制该上游端口中的所有交换机
        + 交换机被允许充当PTM根，或在其上游端口上发出PTM请求以获得PTM主时间，用于履行在其下游端口处接收到的PTM请求。在后一种情况下，交换机必须考虑交换机内的任何内部延迟
        + 交换机被允许维护本地PTM上下文，以用于满足在其下游端口上接收的PTM请求。
        + 不作为PTM根的交换机必须在其上游端口上的最后一个PTM对话后10 ms内使其本地上下文然后，交换机必须在其下游端口上发出进一步的PTM ResponseD消息之前刷新其本地PTM上下文如果通过特定于实现的方式保证交换机的本地时钟与PTM主时间锁相，则此定期刷新要求是可选的
        + 为了维护本地PTM上下文而实施本地时钟的任何交换机必须按照PTM功能结构（[第7.9.16](#_bookmark66)）中的定义报告此时钟的粒度

# 执行说明

事务层逻辑

时间

数据链路层逻辑

Rx时间戳捕获

不对称

Tx时间戳捕获

PTM时间戳捕获机制

PTM使用来自数据链路层和事务层的服务准确性要求尽可能靠近物理层进行时间测量相反，消息传递协议本身属于事务层。PTM消息协议适用于单个链路，其中上游端口是请求者，下游端口是响应者。

*图6-27 PTM时间戳捕获示例*

Tx链路

Rx链路

[图6-27](#_bookmark140)说明了如何选择合适的时间戳捕获点。对于某些实现，事务层和数据链路层内的逻辑是不确定的。实施细节和当前条件对特定数据包何时可能遇到任何特定处理步骤有相当大的影响。这使得如果在更高层中捕获时间戳，则实际上不可能捕获准确记录特定物理事件的时间的任何时间戳

* 1. 准备就绪通知（RN）

准备就绪请求（RN）旨在减少软件在DRS事件或FRS事件之后向设备或功能发出配置请求之前需要等待的时间RN包括设备就绪状态（DRS）和

功能就绪状态（FRS）机制。这些机制提供了简化就绪的直接指示（参见“简化就绪”的术语和首字母缩略词条目当使用时，DRS和FRS允许CRS机制的改进行为，并消除其在重置后长达1秒的相关周期性轮询时间

允许系统软件/固件提供取代FRS和/或DRS机制的机制，但此类软件/固件机制不在本规范的范围内

执行说明

优化配置就绪性

强烈建议系统固件/软件的实施者尽可能避免不必要的延迟强烈建议硬件设计消除或最小化所需的延迟，并充分利用本规范和相关规范中提供的机制来传达所需的延迟（如果有的话）硬件实施者应适当记录实施行为，以使系统固件/软件能够实施最佳行为。

即使有良好的文档，某些情况下可能会出现问题-例如，当需要从根端口配置空间读取时，系统固件如何从设备就绪状态（DRS）机制中受益在这种情况下，需要特定于平台的知识，即根端口支持即时就绪。

* + 1. 设备就绪状态（DRS）

在实施时，DRS必须用于指示在发生以下任何设备级事件（以下称为“DRS事件”）后，设备何时处于“备份就绪”状态

* + - * 退出冷复位
      * 退出热复位、热复位、环回或禁用
      * 退出L2/L3就绪
      * 端口从DL\_Down状态转换为DL\_Up状态的任何其他情况

DRS消息协议要求包括以下内容：

* + - * DRS没有启用或禁用机制。对于支持DRS的下游端口，链路功能2寄存器中的DRS对于支持DRS的上游端口，强烈建议设置Link Capabilities 2寄存器中的DRS Support位明确允许上游端口发送DRS消息，即使DRS支持位为清除。
      * 当逻辑总线上与DRS上行端口相关的所有非VF功能就绪时，在每次DL\_Down到DL\_Up转换之后，DRS上行端口必须发送DRS消息
        + 当[类型0功能](#_bookmark67)处于“准备就绪”状态时，该功能已准备就绪
        + 作为交换机上游端口的[类型1功能](#_bookmark82)在其为转发就绪且其次级总线上的所有功能均为转发就绪时就绪
        + 当不是交换机上游端口的[类型1](#_bookmark82)功能本身为交换就绪时，该功能就绪
      * 在设备发送DRS消息后，除非发生后续DRS事件，否则由该DRS消息指示为“完成就绪”的非VF功能不得返回CRS完成

与实施DRS的交换机相关的其他要求包括：

* + - * 所有端口必须支持DRS功能
      * 在DRS信令控制字段的每个下游端口处的实现
      * 对于出现在交换机下游端口下方的任何物理集成设备，交换机发送的DRS不会指示该设备的配置就绪状态
        + 对于此类设备，DRS的实施和使用独立于交换机

根端口和交换机下游端口的其他要求包括：

DRS消息接收位的实现，其指示DRS消息的接收

执行说明

DRS消息和ACS源验证

允许功能在被分配总线号之前传输DRS消息这些消息将具有总线号为00h的调度员ID如果下游端口启用了ACS源验证，则这些消息（请参阅[6.12.1.1部分](#_bookmark64)）可能会被检测为ACS违规错误。

* + 1. 功能就绪状态（FRS）

在实施时，FRS必须用于指示特定功能在以下任何功能级别事件（以下称为“FRS事件”）发生后处于准备就绪状态

* + - * 功能电平复位（FLR）
      * 完成[D3Hot](#_bookmark102)到[D0](#_bookmark36)的转换
      * PF中VF启用的设置或清除（SR-IOV）

FRS消息协议要求包括以下内容：

* + - * FRS消息的转发器ID必须指示已更改就绪状态的功能（请参见

[第2.2.8.6.4节](#_bookmark53)）

* + - * FRS消息中的FRS原因字段必须指明该功能更改就绪状态的原因
      * 在功能发送FRS消息后，除非发生后续DRS事件或FRS事件，否则指定功能不得返回CRS完成

实施FRS的交换机的其他要求包括：

* + - * 上游端口和所有下游端口必须支持FRS功能
      * 当FRS协议要求时，上行传输[FRS消息](#_bookmark53)的能力[物理功能](#_bookmark107)（[PF](#_bookmark107)）的附加要求包括：
      * 当VF启用或VF禁用过程完成时，能够向上游传输FRS消息实现FRS的根端口和根复杂事件收集器的其他要求包括：
      * 必须实现FRS扩展功能（参见[第7.8.9](#_bookmark57)）
    1. FRS1000

支持FRS的根端口和根复杂事件收集器必须实现FRS SNMP扩展功能（请参见[第7.8.9节](#_bookmark57)）。

对于根端口，FRS消息队列包含由根端口接收或由根端口生成的[FRS消息](#_bookmark53)

对于根复合体事件收集器，FRS消息队列包含由与根复合体事件收集器相关联的RCiEP生成的或由根复合体事件收集器生成的[FRS消息](#_bookmark53)（参见[第7.9.10](#_bookmark83)）

FRS消息队列必须满足以下要求：

* + - * 重置后，FRS消息队列必须为空
      * 对于根端口，当链路变为DL\_Down时，FRS消息队列必须清空
      * [FRS消息](#_bookmark53)必须按接收顺序排队。
      * 如果在接收或内部生成FRS消息时FRS消息队列未满，则必须将该FRS消息输入队列，且FRS消息接收位必须设置为1b。
      * 如果在接收或内部生成FRS消息时FRS消息队列已满，则必须丢弃该FRS消息，FRS消息溢出位必须设置为1b。必须保留预先存在的FRS
      * 最早的FRS消息必须在FRS消息队列寄存器中可见（参见[7.8.9.4部分](#_bookmark0)）。
      * 写入FRS消息队列寄存器必须从队列中删除最旧的元素。
      * 当FRS消息接收或FRS消息溢出从0b转换为1b时，如果使能，则必须生成中断
  1. 增强分配

增强分配（EA）能力是一种可选能力，允许以类型0和类型1配置报头中的BAR和基本/限制机制不可能的方式分配I/O、内存和总线号资源

只允许将EA应用于某些功能，基于PCI配置空间中所见的功能层次结构，以及基于平台环境中功能存在的某些方面（见[图6-28](#_bookmark141)

).

0型功能

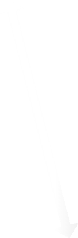
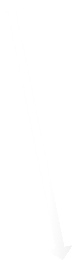
Si组分A

0型功能

总线0位于此虚线

Si组分

B



0型功能

未在配置空间中

类型1功能（桥接）

类型1功能（桥接）

第1类功能

（桥）

此框中的功能必须永久连接到主桥

Si组分C

0型功能

总线N在此虚线

Bus J Bus K



0型功能



0型功能

0型功能

类型1功能（桥接）

公交K+1

0型功能

在辅助侧具有非增强分配（EA）功能的桥接功能被允许在其主总线上使用EA，但不允许在辅助总线



允许在此颜色的函数中使用增强分配

此颜色的函数中不允许增强分配

关键词：

图6-28增强分配

只有永久连接到主桥的功能才允许使用EA。桥接功能（即，当且仅当网桥后面的功能也永久地连接到主桥（在一个或多个网桥下面）时，允许任何具有类型1配置报头的功能在其初级侧和次级侧都使用EA，如图6-28中的“Si组件C”所示

如果网桥后面的功能没有永久连接到网桥，则允许网桥功能仅在其主侧使用EA，如[图6-28](#_bookmark141)中总线J和总线K上面的网桥，并且在这种情况下，总线号、MMIO范围和I/O范围的类型1报头中的非EA资源分配机制用于网桥的次侧系统软件必须确保分配的总线号在EA功能的固定辅助总线号和固定从属总线号寄存器中指示的范围系统软件必须确保分配的MMIO和I/O范围在EA功能中相应属性所指示的范围内，以便在网桥后分配资源对于桥接器后面的总线号、MMIO和I/O范围，允许硬件指示多个桥接器功能中的重叠范围，但是，在这种情况下，系统软件必须确保实际分配的范围不重叠。

专门依赖EA进行I/O和内存地址分配的功能必须将PCI报头中所有BAR的所有位硬连线为0。此类功能必须明确记录为依赖EA进行正确操作，平台集成商必须确保此类功能仅使用EA感知固件/软件

当功能使用EA分配资源并指示资源范围与等效BAR编号相关联时，功能不得通过等效BAR请求资源，这必须通过将等效BAR的所有位硬连线为0来指示

对于基于上述规则被允许实现EA的桥函数， 所述桥接功能使用EA机制来指示位于所述桥接功能之后的资源范围。在[图6-28所](#_bookmark141)示的示例中，允许总线N上的网桥使用EA机制来指示“Si组件C”中的两个功能所使用的资源，或者允许该网桥不指示“Si组件C”中的两个功能所使用的资源。系统固件/软件必须理解，不需要这样的桥功能来指示包括在桥后面的所有资源，并且因此系统固件/软件必须对桥后面的所有功能进行完全搜索以理解由这些功能使用的资源

带有扩展ROM的函数可以使用现有机制或EA机制，但不允许同时支持这两种机制。如果函数使用EA机制（BEI为8的EA条目），则[扩展ROM基址](#_bookmark117)和[扩展ROM启用](#_bookmark31)字段必须硬连线为0（参见[第7.5.1.2.4](#_bookmark208)）。EA条目的使能位相当于[扩展ROM使能](#_bookmark31)位。如果函数使用[扩展ROM基址寄存器](#_bookmark208)机制，则不允许BEI为8的EA条目在这两种情况下，扩展ROM确认（如果支持）使用[扩展ROM确认状态](#_bookmark84)和[扩展ROM确认详细信息](#_bookmark68)字段（参见[第7.5.1.2.4](#_bookmark208)）。

启用和/或禁用I/O和/或内存范围解码的要求不受EA的影响，包括但不限于命令寄存器中的内存空间和I/O空间启用位

使用EA分配的任何资源不得与使用EA分配的任何其他资源重叠，除非上面允许为网桥后面的资源确定允许的地址范围

* 1. 紧急断电状态

紧急功率降低状态是一种可选机制，用于请求功能快速降低其功耗。紧急断电是一种故障安全机制，旨在防止系统损坏，而不是提供正常的动态电源管理。

如果功能实现紧急功率降低状态，则还必须实现功率降低扩展功能，并且必须报告此状态的功率降低值（参见[第7.8.1](#_bookmark27)）。集成在系统主板上的设备不需要实现电源管理扩展功能，但如果要实现，则必须满足上述要求。

函数基于自主或通过外部请求进入和退出此状态外部请求可以遵循在适用的形状因子规范中定义的信令协议，或者通过供应商特定的方法。

[表6-15](#_bookmark143)定义了支持的紧急功率降低和需要的紧急功率降低字段如何确定允许触发进入和退出此状态的机制（参见[第7.5.3.15](#_bookmark13)

).

表6-15紧急功率降低支持值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 支持紧急断电 | 需要紧急断电 | 出入境许可人 | | |
| 外形尺寸机制 | 供应商特定机制 | 自主机制 |
| 00b | 0 | 没有 | 是的 | 是的 |
| 1 | 没有 | 没有 | 没有 |
| 01b | 任何 | 没有 | 是的 | 是的 |
| 10b | 任何 | 是的 | 是的 | 是的 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 支持紧急断电 | 需要紧急断电 | 出入境许可人 | | |
| 外形尺寸机制 | 供应商特定机制 | 自主机制 |
| 11b | 保留 | | | |

函数可以指示它们在退出此状态时需要重新初始化

* 如果紧急功率降低要求位为清除（参见[第7.5.3.15](#_bookmark13)）：
  + 在进入该状态时，功能正常操作（可能具有降低的性能），或者进入设备特定的设备的上游端口保持运行。由职能部门发起或向职能部门发出的未完成请求必须正常完成
  + 退出此状态时，功能正常运行（可能恢复正常性能）。进入“功耗降低休眠状态”的功能在这两种情况下，都不需要软件干预
* 如果设置了紧急功率降低要求位（参见[第7.5.3.15](#_bookmark13)）：
  + 进入此状态后，该功能将停止正常操作。允许关联设备的上游端口进入DL\_Down。
    - 如果上游端口仍处于DL\_Up状态，则指向该功能或由该功能发起的未完成请求必须正常完成。
    - 如果上游端口进入DL\_Down，则未完成的请求行为在[第](#_bookmark9)

[2.9.1](#_bookmark9). 此转换可能会导致Surprise Down错误。

* + - 粘滞位必须在此状态下保留
  + 在退出此状态时，需要软件干预以恢复正常操作。用于在需要时向软件指示的机制在本规范的范围之外（例如，设备特定中断）。如果上游端口进入DL\_Down，则重置设备的所有功能，并需要进行完全重新配置（参见[第2.9.2](#_bookmark41)）。

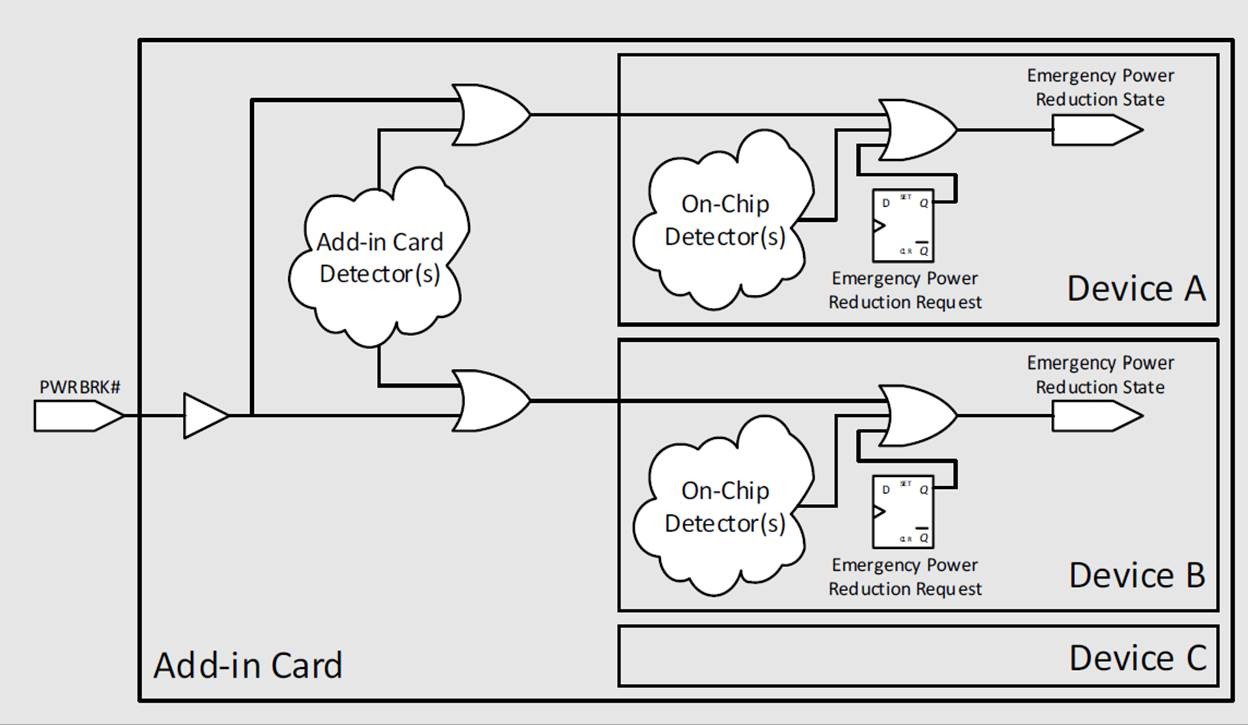
以下规则适用于紧急断电状态：

* 如果上游端口中的至少一个功能指示支持（即，支持的紧急功率降低不为零）。
* 紧急功率降低状态与设备相关联设备中支持它的所有功能同时进入和退出此状态
* 只要上游端口保持在DL\_Up中，支持的紧急功率降低字段为00b的功能就不受设备的紧急功率降低状态的影响紧急功率降低检测位为RsvdZ。
* 支持紧急功率降低字段为01b或10b的功能
  + 当设备进入紧急功率降低状态时，设置检测到紧急功率降低位
  + 如果设备已退出紧急功率降低状态，则在请求时清除检测到紧急功率降低位
* 对于交换机，下游交换机端口与关联的上游交换机端口同时进入和退出紧急功率降低状态配置空间中的相应字段保留用于下游交换机端口。
* 对于SR-IOV设备，VF与它们的PF同时进入和退出紧急功率降低状态配置空间中的相应字段为VF保留
* 除非相关的外形规格规范定义了请求紧急功率降低的机制，否则不得使用编码10b
* 强烈建议在初始设备枚举之前，通过功能内的硬件或固件初始化支持紧急功率降低字段当这不实际时（例如，当没有固件ROM时）。

执行说明

检测到紧急功率降低的诊断检查

紧急功率降低检测位允许系统软件检测是否进入紧急功率降低状态，即使是暂时的。软件可以使用紧急功率降低请求位来请求输入。通常，软件将使用系统特定的方法来使用外部机制进入紧急



执行说明

紧急断电状态：附加卡示例

[图6-29](#_bookmark144)显示了支持紧急断电的多设备附加卡示例注意，设备C不支持紧急功率降低状态。设备C可能是扇出到设备A和B的交换机

图6-29紧急断电状态：附加卡示例

* 1. 层次结构ID消息

当软件对PCI层次结构进行编程时，它为每个组件分配唯一的总线和设备编号，以便层次结构中的每个功能在该层次结构中具有唯一的路由ID为了确保路由ID在包含多个层次结构的大型系统和包含多个层次结构的群集系统中是唯一的，需要附加信息来扩充路由ID以生成唯一的编号。函数可以通过以下组合进行唯一标识

* 系统（或根复合体）
* 该根复合体中层次结构的唯一标识符
* 该层次结构中的路由ID

层次ID消息（参见[第2.2.8.6.5](#_bookmark108)）用于提供函数在多层次平台中唯一标识自身所需的附加信息

分层ID消息由下游端口根据软件请求生成在Hierarchy ID Extended Capability（层次ID扩展能力）中报告上游端口接收到的消息（参见[第7.9.18](#_bookmark103)）。

层次结构ID消息是PCI-SIG定义的类型1 VDM。层次ID消息可以在任何时候安全地发送，不理解它们的组件将默默地忽略它们。

层次ID消息通常从层次顶部的下游端口发送（例如，根端口）。在根端口不支持层次ID消息的系统中，可以从交换机下游端口发送层次ID消息

分层ID消息旨在由软件、固件和/或硬件使用。当使用层次ID消息时，必须比较层次ID、系统权限、系统权限ID字段的所有位，而不考虑任何内部结构。如何使用此信息不在本规范的范围

层次ID消息的布局如[图2-33](#_bookmark69)所示。层次结构ID消息中的字段如下：

**Hierarchy ID**包含与此Hierarchy相关联的段组编号（如*PCI固件规范*所定义）。此字段可与路由ID结合使用，以唯一标识系统内的功能值0000h表示根复合体的默认（或唯一）层次结构。非零值表示其他层次结构。

**系统认证**[143：0]与系统认证机构ID一起为系统提供了一个全球唯一的标识

系统ID [143：136]是层次ID消息中的字节14。系统ID [135：128]是层次ID消息中的字节15。系统ID [127：120]是层次ID消息中的字节16。系统ID [119：112]是层次结构ID消息中的字节17。系统ID [111：104]是层次ID消息中的字节18。系统ID [103：96]是层次结构ID消息中的字节19。系统ID [95：88]是层次结构ID消息中的字节20。系统ID [87：80]是层次ID消息中的字节21。系统ID [79：72]是层次结构ID消息中的字节22。系统ID [71：64]是层次结构ID消息中的字节23。系统ID [63：56]是层次结构ID消息中的字节24。系统ID [55：48]是层次结构ID消息中的字节25。系统ID [47：40]是层次ID消息中的字节26。系统ID [39：32]是层次结构ID消息中的字节27。系统ID [31：24]是层次结构ID消息中的字节28。系统ID [23：16]是层次结构ID消息中的字节29。系统ID [15：8]是层次结构ID消息中的字节30。

系统ID [7：0]是层次结构ID消息中的字节31

**系统权限ID**标识用于确保系统权限在全局上唯一的机制。为给定系统选择使用哪个授权ID的机制是特定于实现定义值见[表6-16](#_bookmark145)。

表6-16系统认证机构ID编码

|  |  |
| --- | --- |
| 授权ID | 描述 |
| 00h | **无**- 系统故障[143：0]没有意义。 |

|  |  |
| --- | --- |
| 授权ID | 描述 |
|  | 系统[143：0]必须为0。 |
| 01h | **Timestamp**- 系统时间戳[63：0]包含与特定系统关联的时间戳编码是Unix 64位时间（从UTC 1970年1月1日午夜开始的秒  选择时间戳来表示系统的机制是特定于实现的系统[143：64]必须为0。 |
| 02h | **IEEE EUI-48**- 系统规范[47：0]包含与特定系统相关的48位扩展唯一标识符（EUI-48）编码由IEEE定义详见[[EUI-48](#_bookmark85)]。EUI-48值经常用作网络接口MAC地址。  选择EUI-48值来表示系统的机制是特定于实现系统[143：48]必须为0。 |
| 03h | **IEEE EUI-64**- 系统规范[63：0]包含与特定系统相关的64位扩展唯一标识符（EUI-64）编码由IEEE定义详见[[EUI-64](#_bookmark86)]。  选择EUI-64值来表示系统的机制是特定于实现系统[143：64]必须为0。 |
| 04h | **RFC-4122 UUID**- System UUID [127：0]包含IETF在[[RFC-4122](#_bookmark7)]中定义的UUID 该定义在技术上等同于[[ITU-T-Rec.- X.667](#_bookmark113)]或[ISO-IEC-9834-8]。    选择UUID值来表示系统的机制是特定于实现系统[143：128]必须为0 |
| 05h | **IPv6地址**- System Address [127：0]包含系统的一个网络接口的唯一IPv6地址选择IPv6值来表示系统的机制是特定于实现  系统[143：128]必须为0。 |
| 06h至7Fh | **保留**- 系统配置文件[143：0]包含唯一值。用于确保唯一性的机制超出了本规范的范围。 |
| 80小时至FFh | **PCI-SIG供应商特定**- 系统认证机构ID值80 h至FFh保留用于PCI-SIG供应商特定用途。System REQ [143：128]包含PCI-SIG分配的供应商ID。  系统编号[127：0]包含由供应商分配唯一编号用于分配数字的机制是特定于实现的。一种可能的机制是使用分配给系统的序列号  用于在这些系统认证机构ID之间进行选择的机制是特定于实现的一种用法是允许供应商定义多达128个不同的128位系统可扩展性方案。 |

执行说明

系统的一致性和稳定性

为了支持系统认证的目的，软件应确保单个系统在任何地方都使用相同的系统认证和系统认证授权ID值

实施者应该仔细考虑他们对系统稳定性值的要求。例如，某些用例可能要求该值在系统重新引导时不更改在这些情况下，如果结果由于硬件故障、系统重新配置或发现算法中的变化/并行性而改变，则拾取与所发现的第一个以太网MAC地址相关联的EUI-48值的机制可能会有问题

执行说明

层次结构ID与设备序列号

器械序列号机制也可用于唯一识别组件（参见[第7.9.3节](#_bookmark70)）。如果涉及与每个组件关联的ROM，则设备序列号可能是解决此问题的更昂贵的解决方案

# 执行说明

虚函数和层次ID

层次ID能力可以由虚拟化中介（VI）来模拟这样做可使VF软件访问此层次结构ID信息。

当VF硬件需要访问此信息时，VF应该实现Hierarchy ID功能。这提供了对VF软件和硬件的访问

在某些情况下，VF应该获得与PF相同的信息在其他情况下，特别是涉及虚拟机迁移的那些情况下，向VF呈现与相关联的PF和与该PF相关联的其他VF不同的层级ID信息可能是适当的

支持以下机制

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | VF层次结构ID功能 | 层次结构ID VF可配置 | 层次结构ID可写 | VF软件有权访问 | VF硬件具有访问权限 | VF层次结构数据  联系我们 |
| 1 | 不存在 | n/a | n/a | 没有 | 没有 | 未模拟 |
| 2 | 是的 | 没有 | 仿真 |
| 3 | 本 | 0b | 0b | 是的 | 是的 | 与PF |
| 4 | 1b | 0b | 是的 | 是的 | 与PF |
| 5 | 1b | 1b | 是的 | 是的 | 由VI |

在机制1中，虚拟化中介不模拟该能力。VF软件和硬件没有访问权限。

在机制2中，虚拟化中介模拟该能力并返回所需的任何层次ID信息VF软件可以访问。VF硬件没有访问权限。

在机制3和4中，VF信息与PF相同，并且从接收到的分层ID消息自动填充VF硬件和软件都可以访问。

在机制5中，VF信息由软件（可能是VI）配置VF硬件和软件都可以访问。

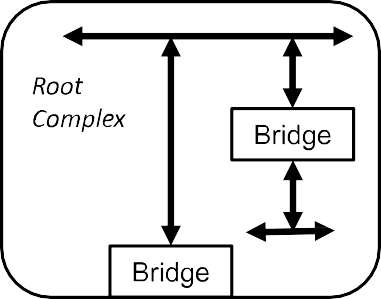
* 1. 扁平门式桥（FPB）
     1. 介绍

扁平化门户桥（FPB）是一种可选机制，可用于提高路由ID和内存空间资源的可伸缩性和运行时重新分配

对于与上游端口相关联的非ARI功能，路由ID由3位功能号部分和13位总线号和设备号部分组成，其中3位功能号部分由上游端口硬件的构造确定，13位总线号和设备号部分由上游端口之上的下游端口确定

对于与上游端口相关联的ARI功能，路由ID由8位功能号部分组成，并且只有8位总线号部分由上游端口之上的下游端口确定

实现[FPB能力](#_bookmark19)的网桥本身也可以称为FPB。[FPB能力](#_bookmark19)可以应用于任何逻辑桥，如[图6-30](#_bookmark147)所示。



*图6-30 FPB高级图和拓扑示例*

FPB通过“扁平化”交换机内部和下游端口使用总线号的方式，改变了交换机使用总线号的方式，以减少浪费

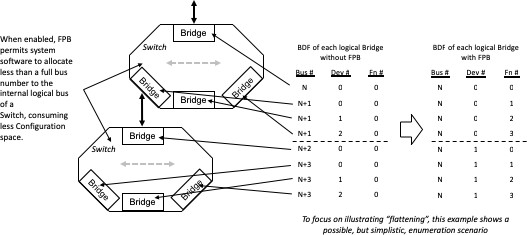
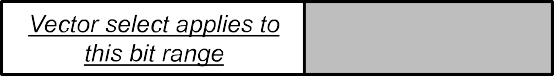


图6-31开关“扁平化”示例

FPB定义了系统软件在非连续范围内分配路由ID和内存空间资源的机制，使系统软件能够分配这些资源的池，从中可以将“bin”分配这是使用位向量来完成的，其中每个位在置位时将相应的资源范围分配给网桥的辅端（参见[图6-32](#_bookmark149)）。





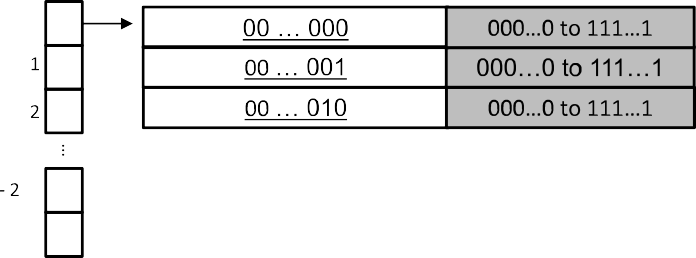
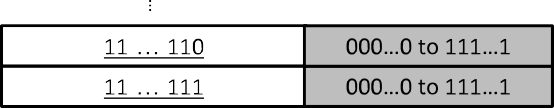
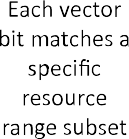


图6-32地址范围解码

这允许系统软件分配设备热添加所需的路由ID和/或存储器空间资源，而不必重新平衡其他已分配的资源范围，并将释放的资源返回到池中，例如通过热移除事件。

FPB被定义为允许非FPB和FPB机制同时运行，例如，系统固件/软件可以实现一个策略，在该策略中，非FPB机制继续用于不需要FPB机制的系统部分（见[图6-33](#_bookmark150)）。在此图中，当给定的TLP被解码为与桥的次级侧相关联时，假设解码逻辑提供非FPB解码机制在没有FPB的情况下也适用，因此，例如，在评估ID路由的TLP时，非FPB解码逻辑仅测试路由ID的总线号部分（位15：8）

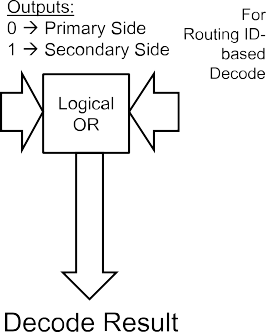
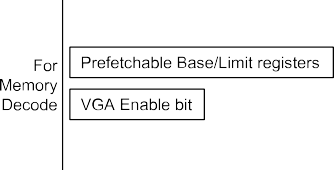
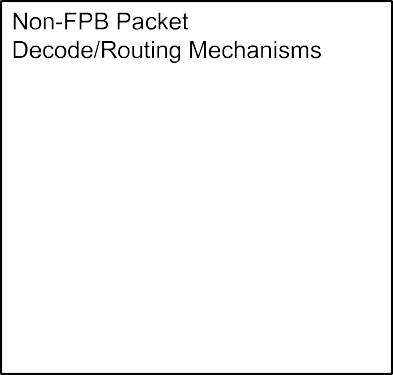


图6-33 FPB和非FPB解码机制

重要的是要认识到，尽管FPB为特定网桥添加了额外的方式来解码给定的TLP，但FPB并没有改变网桥在交换机和根复合体架构结构中运行的基本方式FPB使用相同的架构概念为三种不同的资源类型提供管理机制

1. 路由ID
2. 内存低于4 GB（
3. 内存大于4 GB（

FPB的硬件实现被允许支持这三种机制的任何组合。对于每种机制，FPB使用位向量来针对所选资源类型的特定子集范围指示该范围内的资源是与FPB的主侧相关联还是与辅侧相关联。硬件实现被允许为这些向量实现小范围的大小，并且系统固件/软件被使得能够通过选择应用向量的初始偏移以及向量内的各个比特的粒度来最有效地使用可用向量，以指示给定向量中的比特所应用的资源范围的大小

* + 1. 硬件和软件要求

当使用任何FPB机制时，以下规则适用

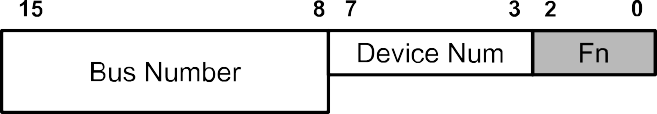
* + - * 如果系统软件违反了任何有关FPB的规则，则硬件行为未定义。
      * 允许在任何PCI桥（类型1）功能中实现FPB，并且实现FPB的每个功能必须实现[FPB能力](#_bookmark19)（参见[第7.8.10](#_bookmark20)）。
      * 如果交换机实现FPB，则交换机的上游端口和所有下游端口必须实现FPB。
      * 允许软件在某些交换机端口上启用FPB，而在其他端口上则不允许。
      * 允许根复合体在某些根端口上实现FPB，但在其他根端口上不允许
      * 允许[类型1功能](#_bookmark82)实现适用于以下任何一种、两种或三种基本机制的FPB机制
        + 路由ID（RID）
        + 内存低于4 GB（
        + 内存大于4 GB（
      * 允许系统软件启用由特定FPB支持的基本机制的任何组合（包括全部或没有）
      * 错误处理和报告机制，除非在本节中明确修改，不受FPB的影响。
      * 在FPB功能的任何复位之后，FPB硬件必须清除所有实现的向量中的所有位
      * 一旦被启用（通过[FPB RID解码机制启用](#_bookmark58)、FPB MEM低解码机制启用和/或FPB MEM高解码机制启用位），如果系统软件随后禁用FPB机制，则相关联的向量中的条目的值是未定义的，并且如果系统软件随后重新启用该FPB机制，则FPB硬件必须清除相关联的向量中的所有位
      * 如果FPB是通过No\_Soft\_Reset位清零实现的，则当FPB通过[D0](#_bookmark36)→[D3Hot](#_bookmark102) →[D0循环](#_bookmark36)时，必须禁用所有FPB机制，并且FPB必须清零所有实现的向量中的所有位
      * 如果FPB是在No\_Soft\_Reset位置位的情况下实现的，则当FPB通过[D0](#_bookmark36)→[D3Hot](#_bookmark102) →[D0](#_bookmark36)循环时，所有FPB配置状态不得改变，并且FPB向量中的条目必须由硬件保留
      * 不需要硬件对FPB计算执行任何类型的边界检查，并且系统软件必须确保FPB参数被正确编程
        + 明确允许系统软件对导致相应矢量的高阶位超过与给定FPB相关联的资源范围的矢量起始值进行编程，但是在这些情况下，系统软件必须确保矢量的那些高阶位是清除的。
        + 系统软件必须避免的错误示例包括重复资源分配、起始偏移量与设置向量位的组合，这些组合可能产生

以下规则适用于FPB路由ID（RID）机制：

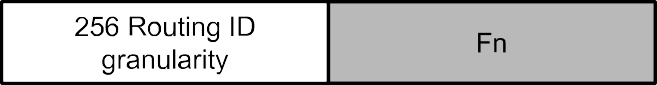
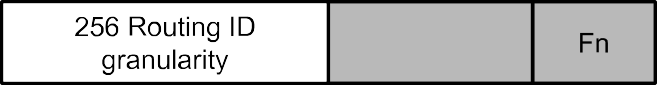
* + - * 如果总线编号部分落在由次级和次级总线编号寄存器中编程的值与FPB RID向量中编程的相应条目中的值进行逻辑或运算所指示的总线编号范围内，则FPB硬件必须考虑与FPB次级侧相关联的RID的特定范围
      * 如果打算仅使用FPB RID机制进行BDF解码，则系统软件必须确保次级和次级总线编号寄存器均为0。
      * 系统软件必须确保配置FPB路由机制，以便FPB将以FPB的功能辅助侧为目标的配置请求从FPB的主要侧路由

当ARI未使能时，FPB RID机制可以不同粒度应用，系统软件可通过[FPB RID矢量控制1寄存器中的FPB RID矢量粒度字段进行编程](#_bookmark14)。 [图6-34](#_bookmark151)说明了RID的布局和支持的粒度之间的关系。读者可能会发现，在考虑下面定义的要求和[压平门式桥（FPB）能力](#_bookmark19)的定义（见[第7.8.6节](#_bookmark87)）时，参考该图是有帮助的





|  |  |  |
| --- | --- | --- |
|  |  |  |

*图6-34路由ID（RID）和支持的粒度*



* + - * 系统软件必须根据[FPB](#_bookmark71)[RID矢量控制1寄存](#_bookmark14)器中的FPB RID矢量粒度和[FPB RID矢量起始](#_bookmark5)字段的描述中描述的约束条件对这些字段进行编程。
      * 对于与交换机上游端口相关的FPB以外的所有
        + 当不支持ARI转发时，或者当[设备控制2寄存器](#_bookmark0)中的[ARI转发使能](#_bookmark58)位被清除时，当设备控制2寄存器的路由ID的位15：3被清除时，FPB硬件必须将在FPB的主侧上接收到的转换为在FPB的辅侧上的类型0配置请求。

类型1配置请求与[FPB RID矢量控制2寄存器](#_bookmark72)中[RID辅助启动](#_bookmark114)字段中的值匹配，系统软件必须相应地配置FPB

* + - * + 当[设备控制2寄存器](#_bookmark0)中的[ARI转发使能](#_bookmark58)位被置位时，当类型1配置请求的路由ID的总线编号部分与[FPB RID矢量控制2寄存器](#_bookmark72)中的次级起始字段的总线编号地址（仅位15：8）中的值匹配时，FPB硬件必须将在FPB初级侧接收的类型1配置请求转换为在FPB次级侧的类型0配置请求，并且系统软件必须相应地配置FPB
      * 对于仅与交换机的上游端口相关联的FPB，当[FPB RID解码机制使能](#_bookmark58)位被置位时，FPB硬件必须使用FPB能力寄存器的[FPB解码器Dev](#_bookmark96)字段来指示与上游端口桥的次级侧相关联的设备号的数量，除了[FPB RID向量控制2寄存](#_bookmark72)器中的[RID次级起始](#_bookmark114)字段之外，FPB还必须使用该数量来确定在FPB的初级侧上接收到的配置请求何时以交换机的下游端口之一为目标，从而实际上确定何时必须将这样的请求从类型1配置请求转换为类型0配置请求，并且系统软件必须适当地配置FPB
        + 配置FPB的系统软件必须理解，交换机的逻辑内部结构将根据交换机的上游端口中的[FPB RID解码机制启用](#_bookmark58)位的值
        + 下游端口必须使用其相应的RID值及其验证器ID和完成器ID，这些值由上游端口的FPB安全开发和RID辅助启动值确定
        + 由交换机上游端口功能的RID辅助启动和FPB辅助启动设备确定的范围内的所有已实现功能

交换机上游端口;需要系统软件扫描此范围内的所有功能，以确定已实施的功能。

* + - * + 强烈建议系统软件分配[RID辅助启动](#_bookmark114)，使总线和设备编号与交换机上游端口的不同;否则，产生的硬件行为未定义。
      * FPB必须为INTx虚拟线路实现网桥映射（参见[2.2.8.1部分](#_bookmark54)）
      * 硬件和软件必须应用此算法（或逻辑等效算法）来确定FPB RID向量中的哪个条目适用于给定的路由ID（RID）地址：
        + 如果RID低于[FPB RID向量开始](#_bookmark5)的值，则RID超出范围（低于开始），因此不能与网桥的次级侧相关联，否则
        + 通过首先减去[FPB RID Vector Start](#_bookmark5)的值，然后根据[FPB RID Vector Granularity](#_bookmark71)的值除以该值来计算向量内的偏移，以确定向量内的位索引
        + 如果位索引值大于由[FPB RID向量大小](#_bookmark118)支持指示的长度，则RID在范围之外（超出向量覆盖的范围的顶部），并且因此不能与桥的次级侧相关联，否则
        + 如果在所计算的位索引位置处的向量内的位值是1b，则RID地址与桥的次级侧相关联，否则RID地址与桥的初级侧相关联。

以下规则适用于FPB MEM低机制：

FPB MEM低电平机制可以以不同的粒度应用，可由系统软件通过[FPB](#_bookmark88) [MEM低电平矢量控制寄存器中的FPB MEM低电平矢量粒度字段进行编程](#_bookmark21)。 [图6-35](#_bookmark152)说明了FPB MEM低机制适用的4 GB以下内存地址空间中地址布局之间的关系。读者可能会发现，在考虑下面定义的要求和[压平门式桥（FPB）能力](#_bookmark19)的定义（见[第7.8.10](#_bookmark20)）时，参考该图是有帮助的



图6-35 4 GB以下内存中的内存占用和粒度

* + - * 系统软件必须根据这些字段描述中描述的约束对[FPB](#_bookmark88)[MEM低矢量控制寄存器](#_bookmark21)中的FPB MEM低矢量粒度和[FPB MEM低矢量开始](#_bookmark10)字段进行
      * 如果某个内存地址落在其他桥接器中编程的值所指示的任何范围内，则FPB硬件必须考虑将该内存地址与FPB的次级侧相关联

存储器解码寄存器（下面列举）与FPB MEM低向量中相应条目中编程的值进行逻辑或运算其他桥接存储器解码寄存器包括：

* + - * + 内存基址/极限寄存器
        + 可预取基址/极限寄存器
        + 桥接控制寄存器中的VGA使能位
        + 增强的分配（EA）能力（如果支持）
        + FPB MEM高机制（如果支持和启用）
      * 硬件和软件必须应用此算法（或逻辑等效算法），以确定FPB MEM低向量中的哪个条目适用于给定的内存地址：
        + 如果存储器地址低于[FPB MEM低矢量开始](#_bookmark10)的值，则存储器地址超出范围（低于），因此不通过该机制与桥的次级侧相关联，否则
        + 通过首先减去[FPB MEM低向量开始](#_bookmark10)的值，然后根据[FPB MEM低向量粒度](#_bookmark88)的值除以该值来计算向量内的偏移，以确定向量内的位
        + 如果位索引值大于由[FPB MEM支持的低向量大小](#_bookmark42)指示的长度，则存储器地址在范围之外（以上），并且因此不借助于该机制与桥的次级侧相关联，否则
        + 如果所计算的位索引位置处的向量内的位值是1b，则存储器地址与桥的次级侧相关联，否则存储器地址与桥的初级侧相关联。

以下规则适用于FPB MEM高机制：

* + - * 系统软件必须根据[FPB MEM](#_bookmark5)[高矢量控制1寄存](#_bookmark59)器中的FPB MEM高矢量粒度和[FPB MEM高矢量起始下限](#_bookmark89)字段的描述中描述的约束条件对这些字段进行编程
      * FPB硬件必须考虑与FPB的次级侧相关联的特定存储器地址，如果该存储器地址落在由编程在其它桥存储器解码寄存器（下面列举）中的值指示的任何范围内，则与编程到FPB MEM高向量中的对应条目中的值进行逻辑或其他桥接存储器解码寄存器包括：
        + 内存基址/极限寄存器
        + 可预取基址/极限寄存器
        + 桥接控制寄存器中的VGA使能位
        + 增强的分配（EA）能力（如果支持）
        + FPB MEM低机制（如果支持和启用）
      * 硬件和软件必须应用此算法来确定FPB MEM高向量中的哪个条目适用于给定的内存地址：
        + 如果存储器地址低于[FPB MEM高向量开始上](#_bookmark73)/[FPB MEM高向量开始](#_bookmark89)下的值，则存储器地址超出范围（低于），因此不通过该机制与桥的次级侧相关联，否则
        + 通过首先减去[FPB MEM高向量开始上](#_bookmark73)/[FPB MEM高向量开始](#_bookmark89)下的值，然后[根据FPB MEM高向量粒度](#_bookmark5)的值除以该值来计算向量内的偏移，以确定向量内的位索引
        + 如果位索引值大于由[FPB MEM支持的高向量大小](#_bookmark90)指示的长度，则存储器地址在范围之外（以上），并且因此不借助于该机制与桥的次级侧相关联，否则
        + 如果所计算的位索引位置处的向量内的位值是1b，则存储器地址与桥的次级侧相关联，否则存储器地址与桥的初级侧相关联。

# 执行说明

FPB地址解码

FPB使用位向量机制来解码路由ID的范围，以及高于和低于4GB的内存位数。支持FPB的网桥包含以下内容，用于支持使用FPB的每个资源类型/范围

* 位向量
* 起始地址
* 的粒度

网桥使用这些来确定给定地址是否是FPB解码的范围的一部分，与网桥的次级侧使用非FPB解码机制和FPB解码机制中的任一者或两者确定为不与桥接器的次级侧相关联的地址（默认地）与桥接器的初级侧相关联这里，当我们使用术语

* 与主服务器关联，在主服务器接收→不支持的请求（UR）
* 与主节点关联，在次节点接收→转发上游
* 与次级相关，在初级接收→向下游
* 与辅助服务器关联，在辅助服务器接收→不支持的请求（UR）

在FPB中，向量中的每个比特表示资源的范围，其中该范围的大小由所选择的粒度确定如果向量中的位被置位，则其指示寻址到对应范围内的地址的TLP将与桥接器的次级侧相关联每个位表示的资源的特定范围取决于该位的索引以及起始地址粒度中的值起始地址表示位向量所描述的最低地址粒度指示由每个位表示的区域的大小向量中的每个连续位应用于后续范围，根据粒度随每个位

例如，考虑使用FPB来描述MEM低范围的桥接器[FPB MEM低矢量开始](#_bookmark10)已设置为FC 0h，表示位矢量描述的范围从地址FC 00 0000开始。[FPB MEM低矢量粒度](#_bookmark88)已设置为0000b，表示每个位表示1 MB范围。

根据这些值，我们可以确定向量的位0表示从FC 000 0000（FC 00 0000-FC 0 F FFFF）开始的1 MB范围，位1表示FC 10 0000-FC 1F FFFF等。

向量中设置为0的位表示该范围不包括在FPB描述的范围在上面的例子中，如果位0为清除，则由于FPB的原因，寻址到FC 00 0000和FC 0F FFFF之间任何位置的数据包都不应路由到网桥的辅助总线

# 执行说明

FPB的硬件和软件注意事项

FPB旨在解决PCI/PCIe架构中与资源分配效率低下有关的一类问题。这些问题可以被分类为

在静态情况下，由于使用了额外的总线号，并且由于交换机和下游端口的PCI/PCIe架构定义而没有使用设备号，因此存在对层次结构大小和端点数量的限制FPB通过“扁平化”路由ID（RID）的使用来解决这类问题，以便交换机和下游端口能够更有效地

对于动态情况，没有FPB，避免重新平衡的“最佳已知方法”是在相关端口或端点上方的网桥中保留大范围的总线号和存储器空间，以便希望可以在预分配的范围内满足任何未来的这导致可能未使用的分配，这使得路由ID问题更糟，并且在资源受限的平台中，即使对于相对简单的情况，这种方法也难以实现，其中，例如，可能具有实现单个端点的附加卡，该附加卡被具有交换机和两个端点的另一附加卡替换，使得尽管仅一个总线的初始分配已经足够，但是初始分配立即被新的附加卡中断。

附加卡。

对于存储器空间，当热插拔端点可能需要分配低于4GB的存储器空间时，预分配方法是有问题的，存储器空间就其性质而言是有限的资源，即使是相对少量的预分配也会快速用完，并且由于多个系统元件对低于4GB的系统地址空间分配提出要求，因此预分配是没有吸引力的

FPB包括用于使能针对路由器ID和存储器空间两者的不连续资源范围分配/重新分配的机制其目的是允许系统软件能够维护可以在运行时分配（和释放回）的资源“池”，而不会像重新平衡所需的那样

为了支持系统软件对FPB的运行时使用，FPB硬件实现应该避免对正在进行的事务引入停顿或其他类型的中断，包括在系统软件正在修改FPB硬件的状态然而，不期望硬件将尝试识别系统软件以确实影响飞行中的事务的方式错误地修改FPB配置的情况正如非FPB机制一样，系统软件的责任是确保系统操作不会因重新配置操作而损坏

没有明确要求系统固件/软件以特定顺序执行FPB机制的启用和/或禁用，然而，应当注意以分层结构实现资源分配操作，使得系统的硬件和软件元件不被破坏或导致故障。

* 1. 重要产品数据（VPD）

重要产品数据（VPD）是唯一定义系统的硬件、软件和微码元素等项目的信息VPD为系统提供有关各种FRU（现场可更换单元）的信息，包括部件号、序列号和其他详细信息。VPD还提供了一种机制，用于存储被监视设备上的性能和故障数据等信息从系统的角度来看，目标是通过从硬件、软件和微码组件读取信息来收集这些信息

附加卡中的VPD支持是可选的，具体取决于制造商。虽然VPD支持是可选的，但由于VPD对附加卡、系统制造商和即插即用具有固有的好处，因此鼓励附加卡制造商提供VPD

访问VPD的机制见[第7.9.19](#_bookmark74)。

PCI Express的VPD与[[PCI-3.0](#_bookmark11)]中的定义相同这个定义，反过来，是基于早期版本的[[PCI](#_bookmark8)]以及[[PLUG-PCI-ISA-1.0a](#_bookmark24)]。

重要产品数据由小型和大型资源数据类型组成

表6-17小资源数据类型标记位定义

|  |  |  |  |
| --- | --- | --- | --- |
| 偏移 | 字段名称 | | |
| 字节0 | 值= 0xxx xyyyb | | |
| 位7 | 小型资源类型 | 0b |
| 位6：3 | 小物品名称 | xxxx |
| 位2：0 | 长度字节 | YY |
| 从1到n | 实际信息 | | |

表6-18大资源数据类型标记位定义

|  |  |  |  |
| --- | --- | --- | --- |
| 偏移 | 字段名称 | | |
| 字节0 | 值= 1xxx xxxxb | | |
| 位7 | 大型资源类型 | 1b |
| 位6：0 | 大型项目名称 | xxxxxxx |
| 字节1 | 数据项位[7：0]的字节长度（lsb） | | |
| 字节2 | 数据项位[15：8]的字节长度（msb） | | |
| 03至n | 实际数据项 | | |

第一个VPD标签是标识符字符串（02h），提供设备的产品名称

一个VPD-R（10 h）标记用作只读关键字的标头VPD-R列表（包括标签和长度）的校验和必须为零。写入只读数据的尝试将作为no-op执行

一个VPD-W（11 h）标签用作读写关键字的报头包含读/写数据的存储组件是一种非易失性设备，在断电时将保留数据

最后一个标签必须是结束标签（0Fh）。

[表6-19](#_bookmark153)显示了典型VPD中使用的资源数据类型标记的一个小示例。

表6-19典型VPD的

|  |  |  |
| --- | --- | --- |
| TAG 02h | 标识符串 | 大型资源数据类型 |
| TAG 10 h | 包含一个或多个VPD关键字的VPD-R列表 | 大型资源数据类型 |
| TAG 11 h | 包含一个或多个VPD关键字的VPD-W列表 | 大型资源数据类型 |
| 标签0Fh | 结束标记 | 小型资源数据类型 |

* + 1. VPD格式

VPD资源类型中的信息字段由一个三字节报头和一些数据组成（请参见

[图6-36](#_bookmark154)）。三个字节的标头包含一个两个字节的关键字和一个一个字节的长度。关键字是唯一标识字段中信息的双字符（ASCII）助记符。头的最后一个字节是二进制的，表示后面数据的长度值（以字节为单位）

字节0 →

+0

+1

+2

+3

7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0

关键字长度（N）

数据（字节3至N+2，图中显示N = 6）

字节4 →

字节8 →

图6-36 VPD格式

VPD关键字分为两类：只读字段和读/写字段。除非另有说明，否则关键字数据字段以ASCII字符形式提供。使用ASCII允许关键字数据在不同的企业计算机系统之间移动，而没有翻译困难。

“附加卡序列号”VPD项的示例

表6-20附加序列卡号

|  |  |  |
| --- | --- | --- |
| 字节0 | 53小时 | 关键词：SN |
| 字节1 | 4Eh |
| 字节3 | 08h | 长度：8 |
| 字节4 | 30小时 | 数据： |
| 字节5 | 30小时 |
| 字节6 | 30小时 |
| 字节7 | 30小时 |
| 字节8 | 30小时 |
| 字节9 | 31小时 |
| 字节10 | 39小时 |
| 字节11 | 34小时 |

* + 1. VPD定义

本节介绍当前的VPD大型和小型资源数据标记以及VPD关键字。此列表可能随时更新。希望定义新关键字的公司应与PCISIG联系所有未指定的值都保留用于SIG分配。

* + - 1. VPD大型和小型资源数据标记

VPD包含在四种类型的大型和小型资源数据标签中。PCI设备中可能会提供以下标签和VPD关键字字段

表6-21 VPD大小资源数据标记

|  |  |
| --- | --- |
| 标签 | 描述 |
| 大资源类型标识符字符串标签（02h） | 此标记是VPD存储组件中的第一项它包含以字母数字字符表示的附加卡名称 |
| 大资源型VPD-R标签（10小时） | 此标记包含附加卡的只读VPD关键字 |
| 大资源类型VPD-W标签（11 h） | 此标记包含附加卡的读/写VPD关键字 |
| 小资源类型结束标签（0Fh） | 此标记标识存储组件中VPD的结尾 |

* + - 1. 只读字段

表6-22 VPD只读字段

|  |  |  |
| --- | --- | --- |
| 关键字 | 名称 | 描述 |
| PN | 附加卡部件号 | 此关键字作为[图6-36](#_bookmark154)中配置空间标题中设备ID（或子系统ID）的扩展提供。 |
| EC | 附加卡工程 | 这些字符是字母数字，表示此附加卡的工程更改级别 |
| FG | 面料地理 | 保留用于[[PICMG](#_bookmark15)]规范的遗留使用 |
| LC | 位置 | 保留用于[[PICMG](#_bookmark15)]规范的遗留使用 |
| MN | 制造商ID | 此关键字作为[图6-36](#_bookmark154)中配置空间标题中的[供应商ID](#_bookmark43)（或[子系统供应商ID](#_bookmark207)）的扩展提供。这使得供应商能够灵活地确定与该设备的采购有关的额外详细信息 |
| PG | PCI  地理 | 保留用于[[PICMG](#_bookmark15)]规范的遗留使用 |
| SN | 序列号 | 这些字符是字母数字，代表唯一的附加卡序列号。 |

|  |  |  |
| --- | --- | --- |
| 关键字 | 名称 | 描述 |
| TR | 热报告 | 该关键字提供了用于报告四个字段的标准界面：***AFI Level***、***MaxTherm***、***DTerm***和***MaxAmbient***。此字段的数据区为四个字节长。此数据按小端顺序编码为4字节二进制值（字节0包含位7：0）。该值包含如下四个字段：[AFI级别](#_bookmark155)位[3：0]、[MaxTherm](#_bookmark155)位[7：4]、[DTherm](#_bookmark155)位[11：8]和[MaxAmbient](#_bookmark156)位[19：12]置于位19：0中位31：20为保留位，必须设置为000h。字段描述在  [[ECN-CEM-THERMAL](#_bookmark109)]至[[CEM-3.0](#_bookmark94)]。此关键字仅用于基于该外形规格的设计中。  请注意，由于VPD编码机制的字符特性，允许此二进制值从VPD内的任何字节边界开始 |
| VX | 供应商特定 | 这是一个供应商特定项目，字符为字母数字。关键字的第二个字符（x）可以是0到9或A到Z。 |
| CP | 扩展能力 | 此字段允许在VPD区域中标识新功能由于动态控制/状态不能放在VPD中，因此此字段的数据标识在设备的存储器或I/O地址空间中可以找到功能的控制/状态寄存器的位置。控制/状态寄存器的位置通过提供定义包含寄存器的地址范围的基址寄存器的索引（0到5之间的值）以及控制/状态寄存器所在的基址寄存器范围内的偏移量此字段的数据区为四个字节长。第一个字节包含扩展能力的ID第二个字节包含所用基址寄存器的索引（从零开始）接下来的两个字节包含为该功能定义的控制/状态寄存器所在地址范围内的偏移量（按小端顺序）。 |
| RV | 校验和和保留 | 此项的第一个字节是校验和字节。如果VPD中的所有字节之和（从VPD地址0到该字节（包括该字节））为零，则校验和正确此项的其余部分是保留空间（根据需要），用于标识只读空间的最后一个字节读写区域没有校验和。此字段为必填字段。 |

* + - 1. 读/写字段

表6-23 VPD读/写字段

|  |  |  |
| --- | --- | --- |
| 关键字 | 名称 | 描述 |
| VX | 供应商特定 | 这是一个供应商特定项目，字符为字母数字。关键字的第二个字符（x）可以是0到9或A到Z。 |
| YX | 系统特定 | 这是一个系统特定项目，字符为字母数字。关键字的第二个字符（x）可以是0到9或B到Z。 |
| 雅 | 资产标签标识符 | 这是一个系统特定项目，字符为字母数字。此关键字包含系统所有者提供的系统资产标识符 |
| RW | 剩余读/写区 | 此描述符用于标识读/写空间的未使用部分。产品供应商根据读/写空间的大小或Vx VPD项之后剩余的空间来确定此参数Vx、Yx和RW项目中的一个或多个是必需的。 |

* + - 1. VPD示例

以下是典型VPD的示例

表6-24 VPD示例

|  |  |
| --- | --- |
| 偏移 | 项目值 |
| 0 | 大资源类型ID字符串标签（02h）82h |
| 1 | 长度0021h |
| 3 | 数据 |
| 36 | 大资源型VPD-R标签（10 h）90 h |
| 37 | 长度0059h |
| 39 | VPD关键字 |
| 41 | 长度08h |
| 42 | 数据 |
| 50 | VPD关键字 |
| 52 | 长度0Ah |
| 53 | 数据 |
| 63 | VPD关键字 |
| 65 | 长度08h |
| 66 | 数据 |
| 74 | VPD关键字 |
| 76 | 长度04h |
| 77 | 数据 |
| 81 | VPD关键字 |
| 83 | 长度2Ch |
| 84 | 数据校验和 |
| 85 | 数据保留（00小时） |
| 128 | 大资源类型VPD-W标签（11 h）91 h |
| 129 | 长度007Ch |
| 131 | VPD关键字 |
| 133 | 长度05h |
| 134 | 数据 |
| 139 | VPD关键字 |
| 141 | 长度0Dh |
| 142 | 数据 |
| 155 | VPD关键字 |

|  |  |
| --- | --- |
| 偏移 | 项目值 |
| 157 | 长度61h |
| 158 | 数据保留（00小时） |
| 255 | 小资源类型结束标记（0Fh）78 h |

* 1. 本机PCIe盘柜管理

NPEM是一种可选的PCIe扩展功能，可提供机箱管理机制此机制旨在为包含PCIe固态硬盘的机箱提供管理，这与存储生态系统中已建立的功能一致。

本节定义了该机制的体系结构方面[第7.9.20](#_bookmark91)节定义了NPEM扩展能力。

盘柜是包含一个或多个PCIe SSD的任何平台、机箱、机架或机箱组。NPEM功能提供与存储相关的存储模块控制（例如，状态LED控制）。NPEM能力可以驻留在下游端口或端点（即，PCIe SSD）。[图6-37](#_bookmark157)显示了具有单个下游端口的示例配置，其中包含NPEM功能和用于控制相关LED的供应商特定逻辑

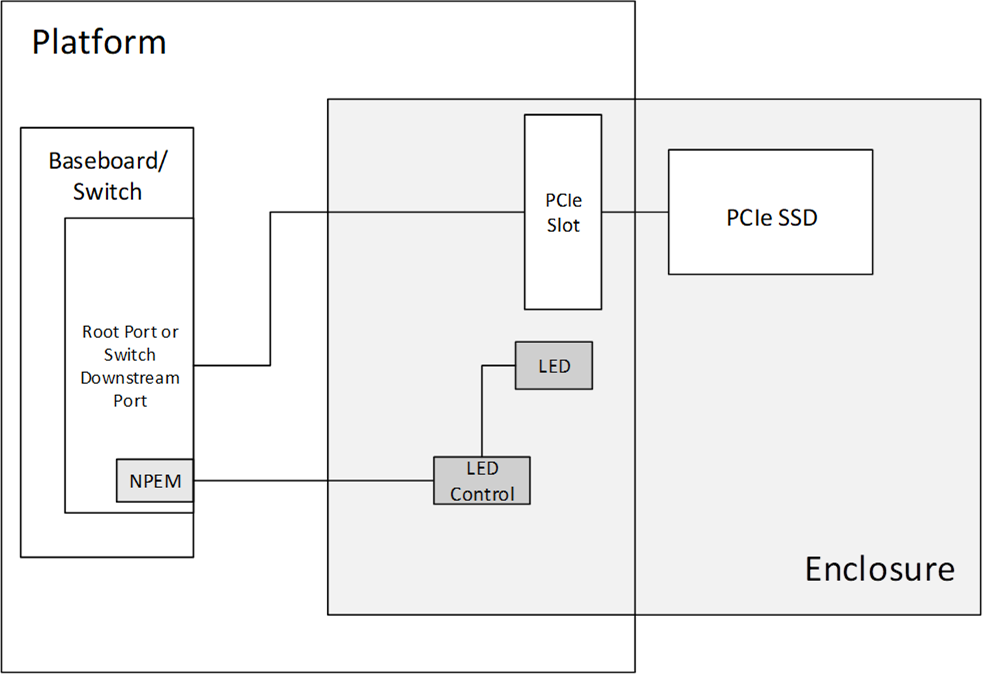


图6-37使用下游端口

[图6-38](#_bookmark158)显示了一个配置示例，其中NPEM功能位于上游端口（在本例中为SSD功能）。

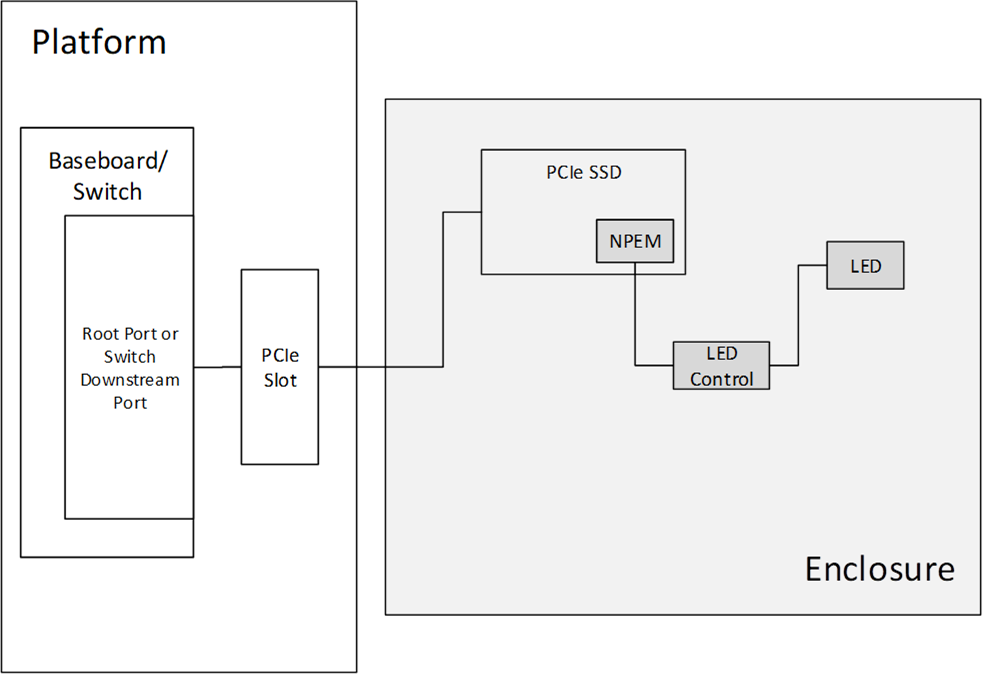


图6-38使用上游端口

软件通过写入NPEM控制寄存器来发出NPEM命令，以更改与SSD相关的指示NPEM命令是对NPEM控制寄存器的单次写入，用于更改零位或多位的状态NPEM使用命令完成机制指示软件的成功完成[图6-39](#_bookmark159)显示了整个流程。

本规范定义了NPEM功能提供的软件接口端口到存储模块接口、存储模块、存储模块到LED接口、每个SSD的LED数量以及相关的LED闪烁模式均不在本规范的范围内。

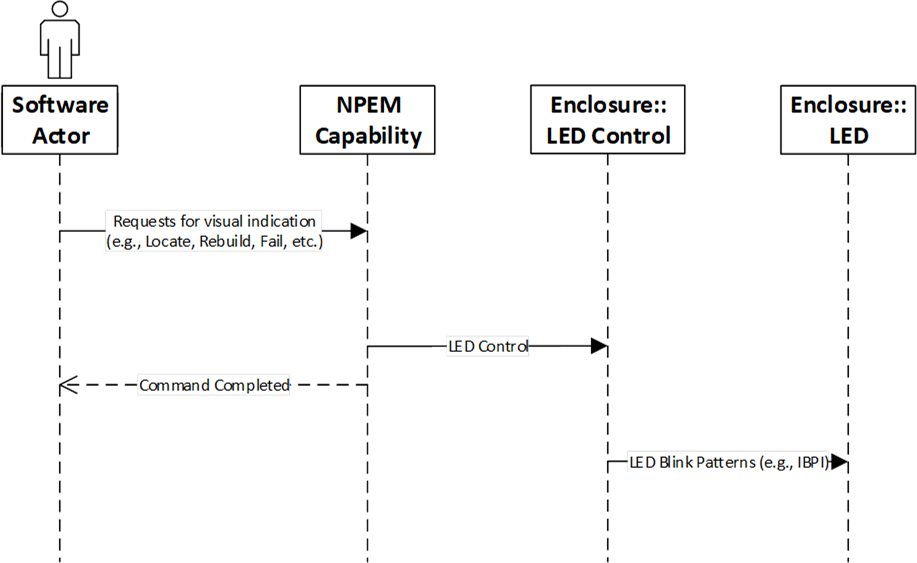


图6-39 NPEM命令流程

NPEM为系统软件提供了一种机制，通过NPEM重置机制向外壳内的LED控制元件发出重置，该机制独立于PCIe链路本身。NPEM命令完成机制也适用于NPEM重置。

存储系统管理员或软件通过NPEM功能控制各种设备状态的指示

# 执行说明

NPEM州

[表6-25](#_bookmark160)显示了NPEM状态的一个示例，以及某些存储模块可能分配给架构NPEM状态的可能含义

表6-25 NPEM状态

|  |  |  |
| --- | --- | --- |
| NPEM  状态 | 演员 | 定义 |
| 好 | 系统管理或存储软件 | 正常状态可能意味着驱动器工作正常。该状态可以隐含地意味着SSD存在、通电并且如软件所看到的那样正常工作驱动器物理上不存在或存在但未通电的更细粒度指示 |
| 定位 | 系统管理 | 定位状态可能意味着管理员正在识别特定的驱动器 |
| 失败 | 存储软件 | 故障状态可能意味着驱动器无法正常工作 |
| 重建 | 存储软件 | 重建状态可能意味着此驱动器是多驱动器存储卷/阵列的一部分，该存储卷/阵列正在将冗余数据重建或重建到此特定驱动器上。 |
| PFA | 存储软件 | PFA是Predicted Failure Analysis的缩写此状态可能意味着驱动器仍正常工作，但预计很快会出现故障。 |
| 热备盘 | 存储软件 | 热备盘状态可能意味着此驱动器被标记为自动用作故障驱动器的替代品，并且故障驱动器的内容可以在此驱动器上重建 |
| 在一个关键的阵列中 | 存储软件 | 处于“关键阵列”状态可能意味着该驱动器是多驱动器存储阵列的一部分，并且该阵列已降级。 |
| 在失败的阵列中 | 存储软件 | NPEM In A Failed Array状态可能意味着驱动器是多驱动器存储阵列的一部分，并且该阵列出现故障。 |
| 无效的设备类型 | 存储软件 | 无效设备类型状态可能意味着驱动器不是连接器的正确类型（例如，盘柜支持SAS和NVMe驱动器，此驱动器状态表示SAS驱动器已插入NVMe插槽）。 |
| 残疾 | 存储软件 | 禁用状态可能意味着此插槽中的驱动器已禁用。从插槽中取出此驱动器可能是安全的。此插槽的电源可能会断开。 |

执行说明

NPEM命令的软件轮询已完成

不同的NPEM实现在完成NPEM命令所需的时间方面可能有很大的不同，从瞬时到数十ms。为了避免或最小化软件轮询开销，建议软件实现以下一种或两种优化。

建议软件颠倒此顺序，而不是软件编写命令，然后立即轮询完成。当准备写入新命令时，软件首先轮询前一个命令的完成，然后写入新命令。这使得重叠操作，通常完全隐藏硬件执行NPEM命令所需的时间。要启用此轮询模型，软件必须在重置后通过写入no-op命令来初始化硬件，以便使硬件生成第一个NPEM命令完成。

对于软件轮询该位时NPEM命令尚未完成的情况，建议软件不要连续“旋转”轮询该位，而是在中断下以较低的速率进行轮询;例如，以10 ms的间隔进行轮询

* 1. 常规PCI高级功能操作

对于集成到根联合体中的常规PCI设备，[常规PCI高级特征能力（AF）](#_bookmark32)提供用于使用最初为PCI Express开发的高级特征的机制

* [功能级复位（INITIATE\_FLR）](#_bookmark115)机制使软件能够以功能级粒度停顿和复位硬件

FLR适用于每个功能。只有目标函数受FLR操作的影响

* [交易未决](#_bookmark119)（TP）机制用于表明该职能部门已发出一个或多个未完成的未过帐交易（包括延迟交易）

这里定义的FLR和TP机制严格适用于集成到根联合体中的常规PCI设备，其中即使其[命令寄存器](#_bookmark178)中的[总线主机使能](#_bookmark182)位的值为0b，该实现也允许给定常规PCI功能的非发布事务完成。不满足此要求的实现不得实现FLR和TP机制。

FLR按如下方式修改功能状态

函数寄存器和函数特定状态机必须设置为本文档中指定的初始化值，但以下位除外，这些位不得修改：[快速背靠背事务启用](#_bookmark185)、[缓存行大小](#_bookmark194)、[延迟计时器](#_bookmark195)、[缓存行](#_bookmark200)、[PME\_En](#_bookmark110)、[PME\_Status](#_bookmark44)。

注意，使功能启动总线事务的控制被清除，包括[命令寄存器](#_bookmark178)中的[总线主机使能](#_bookmark182)位、[MSI能力](#_bookmark75)结构中的[MSI使能](#_bookmark97)位等，有效地使功能变为静止。

启动FLR后，该功能必须在100 ms内完成FLR如果软件在

[交易未决](#_bookmark119)位为1b，则软件不得初始化函数，直到有足够的时间来实现任何未完成交易将已完成的合理确定性事务[挂起](#_bookmark119)位必须在完成FLR后清除。

FLR修改本规范未描述的功能状态（除了本规范描述的状态之外），因此必须使用特定于功能的知识应用以下标准来评估功能

* 该函数不得在该函数控制的任何外部接口上提供具有活动主机的初始化适配器的外观终止外部接口上的活动所需的步骤超出了本规范的范围
  + 例如，网络适配器必须不响应将需要由主机系统进行适配器初始化或与活动主机系统交互的查询，但被允许执行其被设计为在不需要主机初始化或交互的情况下执行的动作如果网络适配器包括在同一外部网络接口上运行的多个功能，则此规则仅影响与FLR重置的特定功能相关联的那些方面
* 该功能不得在其自身内保留软件可读状态，该状态可能包括与该功能的任何先前使用相关的秘密分配给该函数的主机内存不得被该函数修改。
  + 例如，具有可由主机软件直接或间接读取的内部存储器的函数必须清除或随机化该存储器。
* 该功能必须返回到这样一种状态，即该功能的PCI接口的正常配置

启动FLR时，目标函数必须按以下方式运行

* 功能必须正常完成启动FLR操作的配置写入，然后启动FLR。
* 在进行FLR时：
  + 该函数不得响应总线上的任何请求（即以该函数为目标的请求将主中止）。

[事务挂起](#_bookmark119)（TP）位表示该功能已发出一个或多个尚未完成的未过帐事务此字段可由软件用于确定函数何时变为静止。

# 执行说明

避免未决事务的问题

FLR会导致职能部门无法跟踪任何待处理（未结算的未过帐）交易。根据RC集成PCI功能的具体实现，如果软件在存在待处理事务时发出FLR，则可能会出现数据损坏，如“避免陈旧完成导致的数据损坏”实现说明中所述

为了避免可能存在过时完成或存在丢弃计时器的根复合体实现的潜在问题，建议软件使用类似于以下的算法

1. 与可能直接访问函数的其他软件一起执行FLR验证的软件，并确保在此算法期间不会发生此类访问
2. 软件清除整个[命令寄存器](#_bookmark178)，使功能无法控制任何新事务。
3. 软件轮询[AF状态寄存器](#_bookmark12)中的[事务挂起](#_bookmark119)位，直到其被清除或直到其足够长以实现任何剩余未完成事务将永远不会完成的合理确定性。在许多系统上，[事务挂起](#_bookmark119)位通常会在几毫秒内清除，因此软件可能会选择在此初始期间使用紧密的软件循环进行轮询在极少数情况下，当[事务挂起](#_bookmark119)位此时未清除时，软件将需要轮询更长的系统特定时间段（可能为秒），因此软件可能选择使用基于计时器的中断轮询机制进行此
4. 软件启动FLR。
5. 软件等待100 ms。
6. 软件重新配置该功能并使其正常运行。

软件安装和配置

PCI Express配置模型支持两种配置空间访问机制：

* PCI兼容配置访问机制（CAM）（参见[第7.2.1](#_bookmark165)）
* PCI Express增强型配置访问机制（[ECAM](#_bookmark167)）（参见[第7.2.2](#_bookmark166)）

PCI兼容机制支持与传统PCI或更高版本的操作系统及其相应的总线枚举和配置软件的100%二进制兼容性

**第七章**

提供增强的机制以增加可用配置空间的大小并优化访问机制。

* 1. 配置拓扑

为了保持与PCI软件配置机制的兼容性，所有PCI Express元件都具有PCI兼容的配置空间。每个PCI Express链路源自逻辑PCI-PCI桥，并映射到配置空间作为该桥的辅助总线根端口是一个PCI-PCI桥结构，它从PCI Express根复合体中产生PCI Express链路（见[图7-1](#_bookmark162)）。

不使用FPB路由ID机制的PCI Express交换机由多个PCI-PCI桥结构表示，这些PCI-PCI桥结构将PCI Express链路连接到内部逻辑PCI总线（参见[图7-2](#_bookmark163)）。交换机上游端口是一个PCI-PCI桥;此桥的辅助总线代表交换机交换机下游端口是从内部总线桥接到代表来自PCI Express交换机的下游PCI Express链路的总线的PCI-PCI桥接器只有代表交换机下游端口的PCI-PCI网桥才可能出现在内部总线上。端点（由[类型0配置空间标头](#_bookmark202)表示）不允许出现在内部总线上

PCI Express端点作为设备中的单个功能映射到配置空间，该设备可能包含多个功能或仅包含该功能。PCI Express端点和传统端点需要出现在由根复合体发起的层次结构域之一中，这意味着它们出现在以根端口作为其头部的树中的配置空间根复合体集成端点（RCiEP）和根复合体事件收集器不会出现在由根复合体发起的层次结构域之一中这些端口作为根端口的对等端口出现在配置空间

除非另有规定，否则设备配置空间定义中的要求适用于单功能设备以及[多功能设备](#_bookmark116)的每个单独功能。

根复数寄存器块

（可选）

PCI Express根复合体

PCI兼容主机桥接设备

表示根PCI Express端口的

pciexpress链路

图7-1 PCI Express根复合体设备映射

OM14299A

PCI-PCI桥表示上游PCI Express端口

PCI Express交换机PCI-PCI桥

代表下游

PCI Express端口

pciexpress链路

*图7-2 PCI Express交换机设备映射*

OM14300

1. 未来的PCI Express交换机可以实现为单个交换机设备组件（没有PCI-PCI桥），其不受现有PCI软件强加的传统兼容性要求的限制
   1. PCI Express配置机制

PCI Express将每个功能的配置空间扩展到4096字节，而[[PCI](#_bookmark8)]允许的配置空间为256字节PCI Express配置空间分为PCI兼容区域（由功能配置空间的前256个字节组成）和PCI Express扩展配置空间（由剩余配置空间组成）PCI兼容配置空间可以使用

[第7.2.1](#_bookmark165)或[7.2.2节](#_bookmark166)。使用任一访问机制进行的访问是等效的。PCI Express扩展配置空间只能通过使用[第7.2.2](#_bookmark166)中定义的[ECAM](#_bookmark167)机制访问。131

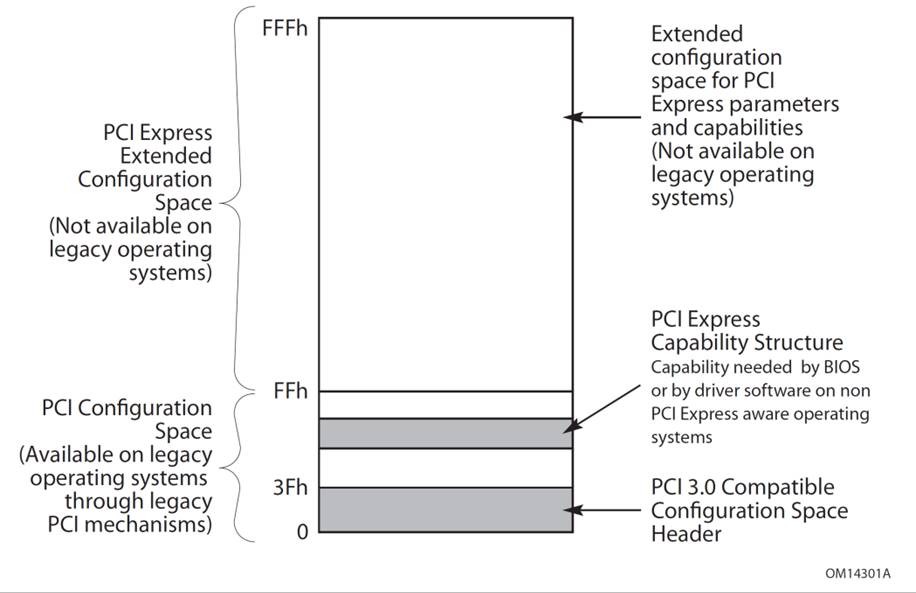


图7-3 PCI Express配置空间布局

* + 1. PCI兼容配置机制

PCI兼容PCI Express配置机制支持[[PCI](#_bookmark8)]中定义的PCI配置空间编程通过遵循此模型，包含PCI Express接口的系统保持与常规PCI总线枚举和配置软件兼容

以与PCI设备功能相同的方式，PCI Express设备功能需要为软件驱动的初始化和配置提供配置除了本章中描述的差异外，PCI Express

1. [第7.2.1](#_bookmark165)中定义的机制和[第7.2.2](#_bookmark166)中定义的[ECAM](#_bookmark167)机制彼此独立地操作;两者之间没有隐含的顺序。

配置空间标头的组织与[[PCI](#_bookmark8)]（[第6.1](#_bookmark25)）中定义的格式和行为相对应

).

PCI兼容配置访问机制使用与[ECAM](#_bookmark167)相同的请求格式。对于PCI兼容配置请求，扩展寄存器地址字段必须全为零。

* + 1. PCI Express增强型配置访问机制（ECAM）

对于与PC兼容的系统，或未实现允许访问配置空间的特定于处理器架构的固件接口标准的系统，需要使用本节中定义的***增强型配置访问机制***（***ECAM***）

对于实现允许访问配置空间的特定于处理器架构的固件接口标准的系统，操作系统使用标准固件接口，并且不需要本节中定义的硬件访问例如，对于符合[DIG64]的系统，操作系统使用SAL固件服务访问配置空间。

在所有系统中，鼓励设备驱动程序使用操作系统提供的应用程序编程接口（API）来访问其设备的配置空间，而不是直接使用硬件机制。

[ECAM](#_bookmark167)利用平坦的存储器映射地址空间来访问设备配置寄存器。在这种情况下，存储器地址决定访问的配置寄存器，存储器数据更新（对于写入）或返回（对于读取）寻址寄存器的内容。[表7-1](#_bookmark168)中定义了从内存地址空间到PCI Express配置空间地址的映射。

映射到配置空间的存储器地址范围的大小和基址由主桥和固件的设计确定它们由固件以特定于实现的方式报告给操作系统。范围的大小由主桥映射到配置地址中的总线号字段的位数决定在[表7-1](#_bookmark168)中，该位数表示为*n*，其中1 ≤*n* ≤ 8。将*n*个存储器地址位映射到总线号字段的主桥支持从0到2*n* -1（包括0和2 n-1）的总线号，并且范围的基址与2（*n*+20）总线号字段中任何未从内存地址位映射的位必须清除。

例如，如果系统将三个内存地址位映射到总线号字段，则以下各项均为真：

* *n*= 3。
* 地址位A[63：23]用于基址，与223字节（8 MB）边界对齐
* 地址位A[22：20]映射到总线号字段中的位[2：0]
* 总线编号字段中的位[7：3]设置为清除。
* 系统能够寻址0和7之间的总线编号（包括0和7）。

至少有一个内存地址位（*n*= 1）必须映射到总线号字段。鼓励系统根据需要将额外的内存地址位映射到总线号字段，以支持更多的总线。鼓励支持4 GB以上内存地址的系统将8位内存地址（*n*= 8）映射到总线号字段。注意，在包括具有分配给每个主桥的不同范围的总线号的多个主桥的系统中，系统的最高总线号受到由最高总线号被分配给的主桥映射的比特数的限制在这种系统中，在大多数情况下，分配给特定主桥的最高总线号将大于分配给该主桥的总线数换句话说，对于每个主桥，映射到总线号字段的位数*n*必须足够大，使得分配给每个特定网桥的最高总线号对于该网桥必须小于或等于*2n*-1

在一些处理器架构中，可能生成不能在单个配置请求中表达的存储器访问，例如由于跨越DW对齐的边界，或者因为使用锁定访问不需要根复合体实现来支持这种访问的配置请求的转换

表7-1增强型配置地址映射

|  |  |
| --- | --- |
| 内存地址132 | PCI Express配置空间 |
| A[（20+*n*-1）：20] | 公交车数1 ≤*n* ≤ 8 |
| [19：15] | 设备号 |
| [14：12] | 功能号 |
| [11：8] | 扩展寄存器编号 |
| A[7：2] | 寄存器编号 |
| A【1：0】 | 与访问大小一起，用于生成字节启用 |

注：对于以ARI设备中的扩展功能为目标的请求，A[19：12]表示（8位）功能编号，它取代了上述（5位）设备编号和（3位）功能编号字段

系统硬件必须为系统软件提供一种方法，以保证使用

[ECAM](#_bookmark167)由完成者在系统软件执行继续之前完成

1. 从软件的角度来看，该地址指的是字节级地址

# 执行说明

增强型配置访问机制的排序注意事项

[ECAM](#_bookmark167)将来自主机CPU的内存事务转换为PCI Express结构上的配置请求。这种转换可能会给软件带来排序问题，因为对存储器地址的写入通常是发布的事务，但对配置空间的写入不会发布在PCI Express结构上

通常，软件不知道完成者何时完成了已发布的事务在这些情况下，软件必须知道一个提交的事务是由完成者完成的，软件通常使用的一种技术是读取刚刚写入的位置。对于始终遵循PCI排序规则的系统，读事务将不会完成，直到提交的写完成。然而，由于PCI排序规则允许非发布的写入和读取事务相对于彼此被重新排序，因此CPU必须等待非发布的写入在PCI Express结构上完成以保证该事务由完成器完成

例如，软件可能希望通过使用ECAM写入器件来配置器件功能的基址寄存器，然后读取该基址寄存器所描述的存储器如果软件在[ECAM](#_bookmark167)写入完成之前发出内存映射读取，则内存映射读取可能会被重新排序并在配置写入请求之前到达设备，从而导致不可预测的结果。

为了避免此问题，处理器和主机桥实现必须确保软件有一种方法来确定完成器何时完成使用[ECAM](#_bookmark167)的写入

该方法可以简单地是处理器本身将专用于映射[ECAM](#_bookmark167)访问的存储器范围识别为唯一的，并且以与处理在PCI Express结构上生成非发布写入的其他访问相同的方式来处理对该范围的访问，即，从处理器的角度来看，交易没有被发布另一种机制是让主桥（而不是处理器）识别

存储器映射的配置空间访问，并且不向处理器指示该写入已经被接受，直到非发布的配置事务已经在PCI Express结构上完成。第三种替代方案是处理器和主机桥将内存映射写入发布到[ECAM](#_bookmark167)，并且主机桥提供一个单独的寄存器，软件可以读取该寄存器以确定配置写入请求何时在PCI Express结构上完成其他替代方案也是可能的。

执行说明

生成配置请求

因为根复合体实现不需要支持从跨越DW边界或使用锁定语义的访问生成配置请求，所以软件在使用存储器映射[的ECAM](#_bookmark167)时应注意不要导致生成这样的访问，除非已知正在使用的根复合体实现将支持转换。

* + - 1. 主桥要求

对于实现[ECAM](#_bookmark167)的那些系统，PCI Express主桥需要将来自主机处理器的存储器映射PCI Express配置空间访问转换为PCI Express配置事务。主桥PCI类代码的使用被保留用于向后兼容性;主桥配置空间对于标准PCI Express软件是不透明的，并且可以以与PCI主桥类型0配置空间兼容的实现特定的方式来实现不需要PCI Express主机桥通过根复合体事件收集器发出错误此支持对于PCI Express主机桥接器是可选的

* + - 1. PCI Express设备要求

器件必须支持额外的4位用于解码配置寄存器访问，即，它们必须解码配置请求报头的扩展寄存器地址[3：0]字段

执行说明

配置空间中的设备特定寄存器

强烈建议PCI Express设备不要在配置空间中放置任何寄存器，除非这些寄存器位于由适用PCI规范构建的标头或功能结构

有正当理由被放置在配置空间中的设备特定寄存器（例如，它们需要在分配内存空间之前可访问）应置于供应商特定能力结构（[第7.9.4](#_bookmark76)节）、供应商特定扩展能力结构（[第7.9.5](#_bookmark42)节或[第7.9.6](#_bookmark46)）中

在运行时环境中由驱动程序访问的特定于设备的寄存器应放置在由一个或多个基址寄存器分配的内存空间尽管PCI兼容或PCI Express扩展配置空间可能有足够的空间用于运行时设备特定寄存器，但出于以下原因，强烈建议将它们放置在那里

* 并非所有操作系统都允许驱动程序直接访问配置空间
* 一些平台仅通过固件调用提供对配置空间的访问，与用于访问存储器空间的机制相比，这通常具有显著较低的性能
* 即使在提供对存储器映射PCI Express增强型配置机制的直接访问的平台上，访问配置空间的性能通常也将显著低于访问存储器空间的性能，因为：
  + 配置读取和写入的大小通常必须为DWORD或更小
  + 配置写入通常不由平台发布，
  + 某些平台一次仅支持一个未完成的配置写入

执行说明

配置空间读取副作用

在读访问期间，除了返回所需值之外发生的任何可观察到的交互都称为读副作用。不具有被访问的函数的特定知识的系统软件可以向函数的配置空间内的任何地方发出读取请求非常不希望任何这样的访问具有任何读取副作用。在本规范中定义的任何配置空间寄存器中不需要这样的副作用强烈建议，这些登记册的任何实施，以及任何

供应商定义的配置空间寄存器，没有任何读取副作用。

* + 1. *根复数寄存器块*

根端口或RCiEP可以与被称为根复合寄存器块（RCRB）的存储器映射寄存器的可选4096字节块相关联这些寄存器以类似于配置空间的方式使用，并且可以包括PCI Express扩展能力和应用于根复合体的其他实现特定的寄存器RCRB的结构描述见[第7.6.2](#_bookmark77)。

允许多个根端口或内部设备与同一RCRB关联RCRB内存映射寄存器不能驻留在与内存映射配置空间或内存空间相同的地址空间

不需要根复合体实现来支持跨DWORD对齐边界的RCRB访问或使用锁定语义的访问

执行说明

复合根寄存器块

由于根复合体实现不需要支持对跨越DW边界或使用锁定语义的RCRB的访问，因此软件在访问RCRB时应注意不要导致生成此类访问，除非根复合体将支持该访问。

* 1. 配置事务规则
     1. 设备号

对于非ARI设备，PCI Express组件仅限于在其主接口（上游端口）上实现单个设备编号，但允许在该设备编号内实现多达8个每个内部功能是基于作为配置请求分组的地址部分

除非使用FPB路由ID机制（参见[第6.27节](#_bookmark146)），否则未启用ARI转发的下游端口必须仅将设备0与连接到逻辑总线（代表来自端口的链路）的设备相针对与指定设备编号0的链路相关联的总线编号的配置请求被发送到连接到该链路的设备;指定所有其他设备编号（1-31）的配置请求必须由交换机下游端口或根端口终止，且请求完成状态为不受支持（相当于PCI中的主机中止

非ARI设备必须捕获其分配的设备编号，如[第2.2.6.2](#_bookmark40)所述。非ARI设备必须响应所有类型0配置读取请求，无论请求中指定的设备编号如何

交换机和希望在其上游端口包含八个以上功能的组件被允许实现一个或多个这些虚拟交换机用于允许扇出超过八个功能。FPB提供了一种由于交换机下游端口允许出现在任何设备编号上，因此在这种情况下，必须对所有地址信息字段（总线、设备和功能编号）进行完全解码，才能访问正确的寄存器。针对未实现的总线、设备或功能的任何配置请求必须返回带有不支持的请求完成状态的完成

对于ARI设备，其设备编号暗示为0，而不是由ID中的字段指定传统的5位设备号和3位功能号字段在其相关的路由ID、验证ID和完成ID中被解释为单个8位功能号。参见[第6.13节](#_bookmark78)。以ARI设备中未实现的功能为目标的任何类型0配置请求必须作为不支持的请求处理

如果ARI下游端口已启用ARI转发，则确定何时将类型1配置请求转换为类型0配置请求的逻辑不再强制将传统设备编号字段限制为0。

以下部分提供配置空间寻址机制的详细信息

* + 1. 配置事务寻址

PCI Express配置请求使用以下地址字段：

* 总线号- PCI Express将逻辑PCI总线号映射到PCI Express链路上，使得PCI兼容配置软件将PCI Express层次结构的配置空间视为包括多个总线段的PCI层次结构
* 器械编号-器械编号关联在[第7.3.1](#_bookmark170)和[第6.27](#_bookmark146)中讨论。当一个ARI设备作为目标，并且它上面的下游端口被启用用于ARI转发时，设备号被暗示为0，并且传统的设备号字段被用作8位功能号字段的一部分参见[第6.13](#_bookmark78)。
* 功能编号-PCI Express还支持使用与PCI相同的发现机制的[多功能设备多功能设备](#_bookmark116)必须完全解码功能编号字段。强烈建议单功能设备也对功能号字段进行完全解码。对于ARI设备，扩展功能的发现和枚举需要ARI感知软件。参见[第6.13](#_bookmark78)。
* 扩展寄存器号和寄存器号-指定正在访问的寄存器的配置空间地址（串联，以便扩展寄存器号形成更高的有效位）。
  + 1. 配置请求路由规则

对于端点，以下规则适用：

* 如果配置请求类型为1，
  + 遵循处理不支持的请求的规则
* 如果配置请求类型为0，
  + 确定请求是否针对已实现功能的有效本地配置空间
    - 如果是，处理请求
    - 如果没有，请遵循处理不支持的请求的规则

对于根端口、交换机和PCI接口-PCI桥接器，以下规则适用：

* 不支持从下游到上游以及对等的配置请求传播
  + 配置请求仅由主桥发起，包括通过[SFICAM](#_bookmark16)机制传递的配置请求
* 如果配置请求类型为0，
  + 确定请求是否针对已实现功能的有效本地配置空间
    - 如果是，处理请求
    - 如果没有，请遵循处理不支持的请求的规则
* 如果Configuration Request Type（配置请求类型）为1，则按顺序对Bus Number（总线号）和Device Number（设备号）字段应用以下测试
  + 如果在PCI接口-PCI桥的情况下，等于分配给辅助PCI总线的总线号，或者在交换机或根联合体的情况下，等于分配给根（根联合体）或下游端口（交换机）之一的总线号和解码设备号，或者如果基于FPB路由ID机制需要
    - 通过更改请求的类型[4：0]字段中的值将请求转换为类型0（参见[表2-3](#_bookmark104)）-请求的所有其他字段保持不变
    - 将请求转发到该下游端口（或PCI总线，如果是PCI接口-PCI桥）
  + 如果不等于任何下游端口或辅助PCI总线的总线号，但是在分配给下游端口或辅助PCI总线的总线号范围内，或者如果基于FPB路由ID机制需要
    - 将请求转发到该下游端口接口，而不进行修改
  + 其他（以上均不）
    - 请求无效-请遵循处理不支持的请求的规则
* PCI扩展-PCI桥接器必须将扩展寄存器地址字段为非零的、指向不支持扩展配置空间的PCI总线的任何配置请求作为不支持的

注：这种类型的访问是编程错误的结果特定于根复合体的附加规则

* 根联合体处理寻址分配给根联合体内的设备的总线号的配置请求
  + 可以以特定于实现方式的方式将总线号分配给根联合体内的设备

对于所有类型的设备：

对未实现的寄存器的配置读取和写入不被视为错误。除非检测到本规范中其他地方定义的错误并需要报告，否则此类请求必须返回具有成功完成状态的完成，其中读取返回全0的数据值，并且写入丢弃写入数据而

所有其他配置空间寻址字段如本说明书中其他地方所述进行解码

* + 1. PCI特殊周期

PCI Express不直接支持PCI特殊周期（详情请参见[[PCI](#_bookmark8)]PCI特殊周期可以被引导到PCI总线段，PCI总线段使用[[PCI](#_bookmark8)]中描述的类型1配置周期

* 1. 配置寄存器类型

配置寄存器字段被分配[表7-2](#_bookmark171)中描述的属性之一。所有PCI Express组件（根复合体和系统集成设备除外）都将寄存器字段初始化为指定的默认值。

根复合体和系统集成设备根据固件的要求为特定的系统实现初始化寄存器字段。

表7-2寄存器和寄存器位字段类型

|  |  |
| --- | --- |
| 注册属性 | 描述 |
| HwInit | **硬件初始化**- 作为一种实现选项，允许对寄存器位进行硬编码、由系统/设备固件初始化或由引脚绑定或非易失性存储等硬件机制初始化133仅允许系统集成设备使用系统固件位必须是固定值，  初始化后只读复位后，仅允许在常规复位后更改值（请参见  [第6.6.1](#_bookmark28)）和随后的重新初始化。HwInit寄存器位不会被FLR修改 |
| RO | **只读**- 寄存器位是只读的，不能被软件改变在明确定义的情况下，这些位用于反映变化的硬件状态，并且因此可以观察到位值在运行时变化。134寄存器位默认值和在运行时不能改变值的位允许被硬编码，由系统/设备固件初始化，或由硬件机制（如引脚绑定或非易失性存储）初始化仅允许系统集成设备使用系统固件  如果未实现设置位的可选功能，则位硬连线至0b。 |
| RW | **读-写**- 寄存器位可读写，允许通过软件设置或清除到所需状态。如果未实现与位相关联的可选特征，则允许位硬连线到0b。 |
| RW1C | **写1-清除状态**- 读取时寄存器位指示状态置位表示通过写入1b而清零的状态事件将0b写入RW1C位不起作用。  如果未实现设置该位的可选功能，则该位为只读并硬连线至0b。 |
| ROS | **粘性-只读**- 寄存器位是只读的，不能被软件改变如果未实现设置该位的可选功能，则该位硬连线至0b。热复位或FLR既不初始化也不修改位135  需要说明的是，当启用辅助功耗（通过[Aux Power PM Enable](#_bookmark105)或[PME\_En](#_bookmark110)）时，消耗辅助功耗的器件必须保留粘性寄存器位值。在这些情况下，寄存器位既不初始化，也不通过热复位、温复位或冷复位进行修改（见[6.6节](#_bookmark1)）。 |

1. 由于历史原因，读者可能会发现本文档中HwInit和RO的使用不一致随着本文件的修订，我们将努力确保新的定义符合此处给出的定义
2. 由于历史原因，读者可能会发现本文档中HwInit和RO的使用不一致随着本文件的修订，我们将努力确保新的定义符合此处给出的定义
3. 必须实现具有“粘性”属性的位/字段，使得不需要功能特定的特别是对于电源管理场景，允许但不建议使用特定功能的软件或固件来恢复正确的值，前提是在系统硬件或系统软件观察到不正确的值之前完成如何做到这一点超出了本文件的范围。

|  |  |
| --- | --- |
| 注册属性 | 描述 |
| RWS | **粘性-读写**- 寄存器位是读写的，并由软件设置或清除为所需状态。热复位或FLR既不初始化也不修改位136  如果未实现与位相关联的可选特征，则允许位硬连线到0b。  需要说明的是，当启用辅助功耗（通过[Aux Power PM Enable](#_bookmark105)或[PME\_En](#_bookmark110)）时，消耗辅助功耗的器件必须保留粘性寄存器位值。在这些情况下，寄存器位既不初始化，也不通过热复位、温复位或冷复位进行修改（见[6.6节](#_bookmark1)）。 |
| RW1CS | **粘滞-写1-清除状态**- 读取时寄存器位指示状态置位表示通过写入1b而清零的状态事件将0b写入RW1CS位无效。如果未实现设置该位的可选功能，则该位为只读并硬连线至0b。热复位或FLR既不初始化也不修改位137  需要说明的是，当启用辅助功耗（通过[Aux Power PM Enable](#_bookmark105)或[PME\_En](#_bookmark110)）时，消耗辅助功耗的器件必须保留粘性寄存器位值。在这些情况下，寄存器位既不初始化，也不通过热复位、温复位或冷复位进行修改（见[6.6节](#_bookmark1)）。 |
| RsvdP | **保留和保留**- 为将来的RW寄存器位是只读的，读取时必须返回软件必须保留读取的值，以便写入位。 |
| RsvdZ | **保留和零**- 为将来的RW 1C实现保留。寄存器位是只读的，读取时必须返回零软件必须使用0b写入位。 |

* 1. 所有端口的基本规格要求的PCI和PCIe功能

以下寄存器和功能是本规范要求的所有功能。

* + 1. PCI兼容配置寄存器

功能配置空间的前256个字节该区域完全混淆了功能的常规PCI配置空间。传统PCI设备也可以通过ECAM访问，而无需对设备硬件或设备驱动程序软件进行任何修改

配置空间的布局和各个配置寄存器的格式按照小端惯例进行描述

* + - 1. 类型0/1公共配置空间

[图7-4](#_bookmark177)详细说明了PCI Express设备功能的类型0和[类型1配置空间报头](#_bookmark55)的公共寄存器字段的标记为特定类型的字段在不同的配置空间标题类型之间有所不同

1. 必须实现具有“粘性”属性的位/字段，使得不需要功能特定的特别是对于电源管理场景，允许但不建议使用特定功能的软件或固件来恢复正确的值，前提是在系统硬件或系统软件观察到不正确的值之前完成如何做到这一点超出了本文件的范围。
2. 必须实现具有“粘性”属性的位/字段，使得不需要功能特定的特别是对于电源管理场景，允许但不建议使用特定功能的软件或固件来恢复正确的值，前提是在系统硬件或系统软件观察到不正确的值之前完成如何做到这一点超出了本文件的范围。

字节偏移

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

设备ID供应商ID

Status命令

修订ID

BIST

类代码

报头类型

等待定时器

高速缓存行大小

类型特异性

能力指针

类型特异性

类型特异性

中断引脚

中断线

+000小时

+004小时

+008小时

+00Ch

+010小时

+014小时

+018小时

+01Ch

+020小时

+024小时

+028小时

+02Ch

+030小时

+034小时

+038小时

+03Ch

图7-4公共配置空间标题

这些寄存器是针对类型0和[类型1配置空间头](#_bookmark55)定义的。本节定义了这些寄存器的PCI专用解释

* + - * 1. *供应商ID寄存器*（偏移00h）

供应商ID寄存器为[HwInit](#_bookmark172)，此寄存器中的值标识函数的制造商为了与PCI-SIG过程保持一致，PCI-SIG必须分配有效的供应商标识符以确保唯一性。每个供应商必须至少有一个供应商ID。建议软件读取供应商ID寄存器以确定是否存在功能，其中FFFFh值表示不存在功能

* + - * 1. *设备ID寄存器*（偏移02h）

设备ID寄存器是[HwInit](#_bookmark172)，该寄存器中的值标识特定的函数。设备ID必须由供应商分配。设备ID与供应商ID和版本ID一起用作软件确定应加载哪个驱动程序的一种供应商必须确保所选的值不会导致使用不兼容的设备驱动程序。

* + - * 1. *命令寄存器*（偏移04h）

[表7-3](#_bookmark181)定义了[命令寄存器](#_bookmark178)，寄存器的布局如[图7-5](#_bookmark180)所示。中的单个位

[命令寄存器](#_bookmark178)可以实现也可以不实现，具体取决于函数支持的特性集对于PCI Express到PCI/PCI-X桥接器，请参阅[[PCIe-to-PCI-PCI-X-Bridge](#_bookmark111)]了解此寄存器的要求

I/O空间使能存储空间使能总线主机使能特殊周期使能

15

11 10 9 8 7 6 5 4 3 2 1 0

RsvdP

内存写入和无效VGA监听

奇偶校验错误响应

IDSEL步进/等待周期控制SERR#启用

快速背对背事务启用禁用

图7-5命令寄存器

表7-3命令寄存器

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 0 | ***I/O空间启用***-控制函数对I/O空间访问的响应。当此位为清除时，所有收到的I/O访问都将作为不支持的请求处理。当此位置1时，功能被启用，以解码地址并进一步处理I/O空间访问。对于具有  [类型1配置空间报头](#_bookmark55)，该位控制对在其主侧接收的I/O空间访问的响应  该位的默认值为0b。  如果功能不支持I/O空间访问，则允许将此位硬连线至0 b | [RW](#_bookmark174) |
| 1 | ***内存空间启用***- 控制函数对内存空间访问的响应当此位为清除时，所有接收到的内存空间访问都将作为不支持的请求处理当此位置1时，功能被启用，以解码地址并进一步处理内存空间 | [RW](#_bookmark174) |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
|  | 访问。对于具有[类型1配置空间报头](#_bookmark55)的功能，此位控制对在其主侧接收到的存储器空间访问的响应  该位的默认值为0b。  如果功能不支持存储空间访问，则允许将此位硬连线至0b |  |
| 2 | ***总线主机使能***- 控制功能发出内存138和I/O读/写请求的能力，以及端口在上游方向转发内存和I/O读/写请求的能力   * 具有[类型0配置空间标题的](#_bookmark202)函数：   当此位置1时，允许功能发出内存或I/O请求。  当此位为清零时，功能不允许发出任何内存或I/O请求。  请注意，由于MSI/MSI-X中断消息是带内存储器写入，因此将[总线主机使能](#_bookmark182)位设置为0 b也会禁用MSI/MSI-X中断消息  除内存或I/O请求以外的请求不受此位控制该位的默认值为0b。  如果功能不生成内存或I/O请求，则此位硬连线至0 b   * 带有Type 1空格头的函数：   此位控制上游方向端口的内存或I/O请求转发当此位为0 b时，在根端口或交换机端口的下游侧接收的内存和I/O请求必须作为不支持的请求（UR）处理，对于非发布请求，必须返回具有UR完成状态的完成此位不影响上游或下游方向的完成  此位不控制除内存或I/O请求以外的请求的转发该位的默认值为0b。 | [RW](#_bookmark174) |
| 3 | ***特殊周期使能***- 此位最初在[[PCI](#_bookmark8)]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 4 | ***内存写入和无效***- 此位最初在[[PCI](#_bookmark8)]和[PCI-to-PCI-Bridge]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。对于PCI Express到PCI/PCI-X桥接器，请参阅[[PCIe-to-PCI-PCI-X-Bridge](#_bookmark111)]了解此寄存器的要求 | [RO](#_bookmark173) |
| 5 | ***VGA监听***- 此位最初在[[PCI](#_bookmark8)]和[PCI-to-PCI-Bridge]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 6 | ***奇偶校验错误响应***- 参见[第7.5.1.1.14](#_bookmark201)。  此位控制在[状态寄存器](#_bookmark186)的[主数据奇偶校验错误](#_bookmark189)位中记录中毒TLP。  不与[根复合事件收集器](#_bookmark47)相关联的[RCiEP](#_bookmark112)被允许将该位硬连线到0b。  该位的默认值为0b。 | [RW](#_bookmark174) |
| 7 | ***IDSEL步进/等待周期控制***- 此位最初在[[PCI](#_bookmark8)]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 8 | ***SERR#启用***- 参见[第7.5.1.1.14](#_bookmark201)。 | [RW](#_bookmark174) |

1. [设备控制2](#_bookmark0)寄存器中的[AtomicOp编译器使能](#_bookmark80)位也必须置位，以便AtomicOp编译器启动AtomicOp请求，即内存请求。

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
|  | 设置时，该位使能向上游报告功能检测到的非致命和致命错误请注意，如果通过此位或[设备控制寄存器](#_bookmark60)中的PCI Express特定位启用，则会报告错误（请参见[第7.5.3.4](#_bookmark60)）。  此外，对于具有[类型1配置空间报头](#_bookmark55)的功能，此位控制从辅助接口转发的[ERR\_NONFATAL](#_bookmark92)和[ERR\_FATAL](#_bookmark98)错误消息的主接口传输此位不影响转发的[ERR\_COR](#_bookmark65)消息的传输  不与[根复合事件收集器](#_bookmark47)相关联的[RCiEP](#_bookmark112)被允许将该位硬连线到0b。  该位的默认值为0b。 |  |
| 9 | ***快速背靠背事务启用***- 此位最初在[[PCI](#_bookmark8)]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 10 | ***中断禁用***- 控制函数生成INTx仿真中断的能力置位时，禁止功能断言INTx中断。  当此位置1时，功能已置位的任何INTx仿真中断必须解除置位  如[第2.2.8.1](#_bookmark54)所述，INTx中断使用虚拟线，如果置位，则必须在此位置位时使用相应的Deassert\_INTx消息  只有与此位被设置的功能相关的INTx虚拟线路中断才会受到影响。  对于具有[类型0配置空间报头](#_bookmark202)并生成INTx中断的功能，需要此位。对于不产生INTx中断的具有[类型0配置空间报头](#_bookmark202)的功能，此位为可选位。如果未实现，此位必须硬连线至0b。  对于具有[类型1配置空间报头](#_bookmark55)并代表自身生成INTx中断的功能，需要此位此位对从次级侧转发的中断没有影响  对于具有[类型1配置空间报头](#_bookmark55)的函数，该位是可选的，该函数不代表其自身生成INTx中断如果未实现，此位必须硬连线至0b。  该位的默认值为0b。 | [RW](#_bookmark174) |

* + - * 1. *状态寄存器*（偏移06 h）

[表7-4](#_bookmark188)定义了[状态寄存器](#_bookmark186)，寄存器的布局如[图7-6](#_bookmark187)所示。函数可能不需要实现所有位，这取决于函数支持的功能集对于PCI Express到PCI/PCI-X桥接器，请参阅[[PCIe-to-PCI-PCI-X-Bridge](#_bookmark111)]了解此寄存器的要求

立即就绪RsvdZ

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

支持66 MHz的RsvdZ

支持快速背对背事务处理的主数据奇偶校验错误

DEVSEL定时信号目标中止接收目标中止接收主机中止信号系统错误检测到奇偶校验错误

图7-6状态寄存器

表7-4状态寄存器

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 0 | ***立即就绪***- 此可选位设置时，表示功能保证随时准备好成功完成有效的配置请求允许这种指示基于主机准备发出配置请求需要多长时间的特定于实现的知识  当该位置1时，对于访问该功能，软件免除任何类型复位后延迟配置访问的所有要求，包括但不限于[第6.6节](#_bookmark1)中定义的时序要求。  如何建立这一保证超出了本文件的范围  允许系统软件/固件提供取代该位提供的指示的机制，但此类软件/固件机制不在本规范的范围 | [RO](#_bookmark173) |
| 3 | ***中断状态***- 设置时，表示INTx仿真中断在函数内部挂起  请注意，此位不反映从次侧通过具有[类型1配置空间报头](#_bookmark55)的函数转发的INTx仿真中断  设置禁用位对该位的状态没有影响  允许不生成INTx中断的函数将此位硬连接到0b。该位的默认值为0b。 | [RO](#_bookmark173) |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 4 | ***功能列表***- 指示存在扩展功能列表项。由于所有PCI Express设备功能都需要实现PCI Express能力结构，因此该位必须硬连线至1b。 | [RO](#_bookmark173) |
| 5 | ***66 MHz能力***- 此位最初在[[PCI](#_bookmark8)]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 7 | ***支持快速背靠背事务***- 此位最初在[[PCI](#_bookmark8)]中描述它的功能不适用于PCI Express，并且该位必须硬连接到0b。 | [RO](#_bookmark173) |
| 8 | ***主数据奇偶校验错误***- 参见[第7.5.1.1.14](#_bookmark201)  如果[命令寄存器](#_bookmark178)中的[奇偶校验错误响应](#_bookmark183)位为1b，并且出现以下两种情况之一，则该位由具有[类型0配置空间报头](#_bookmark202)的函数置1   * 函数接收中毒完成 * 函数发送中毒请求   如果[命令寄存器](#_bookmark178)中的[奇偶校验错误响应](#_bookmark183)位为1b，并且出现以下两种情况之一，则该位由具有[类型1配置空间报头](#_bookmark55)的函数置位   * 端口接收到下游的中毒完成 * 端口向上游传输中毒请求   如果[奇偶校验错误响应](#_bookmark183)位为0b，则此位从不置位。该位的默认值为0b。 | [RW1C](#_bookmark175) |
| 十点九分 | ***DEVSEL Timing***- 此字段最初在[[PCI](#_bookmark8)]中描述其功能不适用于PCI Express，并且该字段必须硬连线到00b。 | [RO](#_bookmark173) |
| 11 | ***发出目标中止***信号-参见[第7.5.1.1.14](#_bookmark201)。  当功能完成已发布或未发布请求时，此位被设置为完成程序中止错误。当完成程序中止由其主端生成时，这适用于具有[类型1配置空间标题](#_bookmark55)的函数  允许具有[类型0配置空间报头](#_bookmark202)的函数将此位硬连接到0b，该函数不发出完成程序  该位的默认值为0b。 | [RW1C](#_bookmark175) |
| 12 | ***收到目标中止***-参见[第7.5.1.1.14节](#_bookmark201)。  当完成程序收到完成程序中止完成状态的完成时，该位置1在具有[类型1配置空间报头](#_bookmark55)的函数上，当其主侧接收到完成器中止时，该位被置位  具有[0型配置空间报头](#_bookmark202)的函数，如果不代表自己发出非发布请求，则允许将此位硬连接到0b。  该位的默认值为0b。 | [RW1C](#_bookmark175) |
| 13 | ***收到主中止***-参见[第7.5.1.1.14节](#_bookmark201)。  当应答器收到一个带有不支持的请求完成状态的完成时，此位置1在具有[类型1配置空间报头](#_bookmark55)的功能上，当其主侧接收到不支持的请求时，该位被置位  具有[0型配置空间报头](#_bookmark202)的函数，如果不代表自己发出非发布请求，则允许将此位硬连接到0b。  该位的默认值为0b。 | [RW1C](#_bookmark175) |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 14 | ***信号系统错误***- 参见[第7.5.1.1.14](#_bookmark201)。  当功能发送[ERR\_FATAL](#_bookmark98)或[ERR\_NONFATAL](#_bookmark92)消息，且[命令寄存器](#_bookmark178)中的[SERR#使能](#_bookmark184)位为1b时，该位置1  不发送[ERR\_FATAL](#_bookmark98)或[ERR\_NONFATAL](#_bookmark92)消息的具有[类型0配置空间报头](#_bookmark202)的函数允许将此位硬连接到0b。  该位的默认值为0b。 | [RW1C](#_bookmark175) |
| 15 | ***检测到奇偶校验错误***- 参见[第7.5.1.1.14](#_bookmark201)。  无论[命令寄存器](#_bookmark178)中奇偶[校验错误响应](#_bookmark183)位的状态如何，只要函数收到中毒TLP，该位就会置1。在具有[类型1配置空间报头](#_bookmark55)的函数上，当其主侧接收到中毒的TLP时，该位被置位  该位的默认值为0b。 | [RW1C](#_bookmark175) |

* + - * 1. *版本ID寄存器*（偏移08h）

[修订ID寄存器](#_bookmark190)是[HwInit](#_bookmark172)，该寄存器中的值指定函数特定的修订标识符。该值由供应商选择零是可接受的值。设备ID与供应商ID和版本ID一起用作软件确定应加载哪个驱动程序的一种机制供应商必须确保所选的值不会导致使用不兼容的设备驱动程序。

* + - * 1. *分类代码寄存器*（偏移09h）

[类代码寄存器](#_bookmark191)是只读的，用于标识函数的通用操作，在某些情况下，还用于标识特定的寄存器级编程接口。寄存器布局如[图7-7](#_bookmark192)所示，并在[表7-5](#_bookmark193)中进行了描述。

在[[PCI-Code-and-ID](#_bookmark48)]中提供了基类、子类和编程接口的编码保留所有未指定的编码

编程接口

23

16 15

8 7

0

基类代码子类代码

图7-7类代码寄存器

*表7-5类代码寄存器*

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 七比零 | ***编程接口***- 此字段标识特定的寄存器级编程接口（如果有），以便器件独立软件可以与功能交互  此字段的编码在[[PCI-Code-and-ID](#_bookmark48)]中提供保留所有未指定的编码 | [RO](#_bookmark173) |
| 十五点八分 | ***子类代码***- 指定基类子类，更具体地标识函数的操作 | [RO](#_bookmark173) |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
|  | 子类的编码在[[PCI-Code-and-ID](#_bookmark48)]中提供保留所有未指定的编码 |  |
| 二十三点十六分 | ***基类代码***- 对函数执行的操作类型进行广泛分类的代码  基类的编码在[[PCI-Code-and-ID](#_bookmark48)]中提供保留所有未指定的编码 | [RO](#_bookmark173) |

* + - * 1. ***高速缓存行大小寄存器*（偏移0Ch）**

缓存行大小寄存器由系统固件或操作系统编程为系统缓存行大小。但是，请注意，传统的PCI兼容软件可能并不总是能够正确编程此寄存器，特别是在热插拔设备的情况下。此读写寄存器的实现是为了实现传统兼容性，但对任何PCI Express设备行为没有影响。对于PCI Express到PCI/PCI-X桥接器，请参阅[[PCIe-to-PCI-PCI-X-Bridge](#_bookmark111)]了解此寄存器的该寄存器的默认值为00h。

* + - * 1. *延迟定时器寄存器*（偏移0Dh）

此寄存器也称为[第1类配置空间报头](#_bookmark55)功能的主延迟定时器延迟计时器最初在[[PCI](#_bookmark8)]和[PCI-to-PCI-Bridge]中描述它的功能不适用于PCI Express。该寄存器必须硬接线至00h。

* + - * 1. *标题类型寄存器*（偏移量0Eh）

该寄存器标识预定义标头第二部分的布局（从配置空间中的字节10 h开始），以及设备是否可能包含多个功能。寄存器布局如[图7-8](#_bookmark196)所示，[表7-6](#_bookmark197)描述了寄存器中的位。

多功能装置

7 6

0

页眉布局

图7-8报头类型寄存器

表7-6报头类型寄存器

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 六比零 | ***页眉布局***- 此字段标识预定义页眉的第二部分的布局  对于实现[类型0配置空间报头](#_bookmark202)的函数，必须使用编码000 0000b  对于实现[类型1配置空间报头](#_bookmark55)的函数，必须使用编码000 0001b | [RO](#_bookmark173) |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
|  | 编码000 0010b是保留的。这种编码最初在[[PC-Card](#_bookmark33)]中描述，并在编程模型的早期版本应认真考虑任何改变其用途的企图  所有其他编码均保留。 |  |
| 7 | ***多功能设备***-设置时，表示设备可能包含多个功能，但不一定。允许软件探测功能0以外的功能清除时，软件不得探测除功能0以外的功能，除非另一机制（如ARI或SR-IOV扩展功能结构）明确指示除非另有说明，否则如果有多个功能，建议将该位置1，如果只有一个功能，建议将该位清零 | [RO](#_bookmark173) |

* + - * 1. *BIST寄存器*（偏移0 Fh）

此寄存器用于BIST的控制和状态不支持BIST的函数必须将寄存器硬连接到00h。被调用BIST的函数不得阻止PCI Express链路的正常操作[表7-7](#_bookmark199)描述了寄存器中的位，[图7-9](#_bookmark198)显示了寄存器布局。

完成代码RsvdP

7 6 5 4 3

0

启动BISTBIST Capable

图7-9 BIST寄存器

表7-7 BIST寄存器

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 三比零 | ***完成代码***- 此字段对最近测试的状态进行值为0000b表示函数已通过测试。非零值表示函数失败。功能特定的故障代码可以用非零值编码  此字段此字段的默认值为0000b。  如果BIST Capable为Clear，则此字段必须硬连线至0000b | [RO](#_bookmark173) |
| 6 | ***启动BIST***- 如果BIST Capable已设置，则将此位设置为调用BIST。BIST完成时，该功能复位该位如果该位在设置后  将此位写入0b没有任何影响。  如果BIST Capable为Clear，则此位必须硬连线至0b | RW/RO  (see描述） |

|  |  |  |
| --- | --- | --- |
| 比特位置 | 寄存器描述 | 属性 |
| 7 | ***BIST Capable***- 设置时，此位表示功能支持BIST。清除时，该函数不支持BIST。 | [HwInit](#_bookmark172) |

* + - * 1. *能力指针*（偏移34 h）

此寄存器用于指向此函数实现的功能的链接列表由于所有PCI Express功能都需要实现[PCI电源管理能力](#_bookmark26)和[PCI Express能力](#_bookmark22)结构，因此这些结构必须包含在链接列表中的某个位置;该寄存器可以指向这些能力结构中的任一个或指向由功能实现的可选能力结构底部的两位是保留位，必须设置为00b。软件在将此寄存器用作配置空间中指向新功能链表第一个条目的指针之前，必须屏蔽这些位

* + - * 1. *线路寄存器*（偏移3通道）

中断线寄存器用于传递中断线路由信息。该寄存器为读/写寄存器，必须由使用中断引脚的任何函数实现（见以下描述）。该寄存器中的值由系统软件编程，并且是系统架构特定的。函数本身并不使用此值;而是由设备驱动程序和操作系统使用此寄存器中的值

* + - * 1. *引脚寄存器*（失调3Dh）

PIN寄存器是一个只读寄存器，用于标识函数使用的传统中断消息（请参见

[第6.1](#_bookmark25)了解更多详情）。有效值为01h、02h、03h和04h，分别映射到INTA、INTB、INTC和INTD的传统中断消息值00h表示函数不使用传统中断消息。值05h到FFh被保留。

PCI Express为单个功能设备定义了一个传统中断消息，为[多功能设备](#_bookmark116)定义了多达四个传统中断消息。对于单功能设备，只能使用INTA

[多功能设备](#_bookmark116)上的任何功能都可以使用任何INTx消息。如果设备实现单个传统中断消息，则它必须是INTA;如果它实现两个传统中断消息，则它们必须是INTA和INTB;依此类推。对于[多功能设备](#_bookmark116)，所有功能都可以使用相同的INTx消息，或者每个功能都可以有自己的（最多四个功能）或其任何组合。一个单一的功能永远不能产生一个中断请求在一个以上的INTx消息。

* + - * 1. 错误寄存器

命令和状态寄存器中的错误控制/状态寄存器位（分别参见[第7.5.1.1.3](#_bookmark179)和[第7.5.1.1.4](#_bookmark186)）以及[类型1配置空间报头](#_bookmark55)功能的桥接控制和辅助状态寄存器（分别参见[第7.5.1.3.10](#_bookmark17)和[第7.5.1.3.7](#_bookmark61)）控制PCI和PCI Express设备功能的PCI兼容错误报告将PCI Express错误映射到PCI错误也在[第6.2.7.1](#_bookmark56)中讨论。除了PCI兼容的错误控制和状态，PCI Express错误报告可以通过[第7.5.3](#_bookmark22)中描述的PCI Express能力结构与PCI设备功能分开控制。PCI兼容的错误控制和状态寄存器字段对通过PCI Express功能结构启用的PCI Express错误报告没有任何影响PCI Express设备功能可以实现可选的高级错误报告，如[第7.8.4](#_bookmark93)所述。

对于由[类型1配置空间报头](#_bookmark55)表示的PCI Express根端口：

初级侧错误控制/状态寄存器适用于在与根复合体相关的内部逻辑上检测到的错误

辅助端错误控制/状态寄存器适用于在源自根端口的链路上检测到的错误

对于由类型[1配置空间报头](#_bookmark55)表示的PCI Express交换机上游端口：

初级端错误控制/状态寄存器适用于在交换机上行链路上检测到的错误

次级侧错误控制/状态寄存器适用于在交换机内部逻辑上检测到的错误

对于由类型[1配置空间报头](#_bookmark55)表示的PCI Express交换机下游端口：

原边错误控制/状态寄存器适用于在交换机内部逻辑上检测到的错误

次级侧错误控制/状态寄存器适用于在下游链路上检测到的源自交换机端口的错误

* + - 1. 类型0配置空间标题

[图7-10](#_bookmark203)详细说明了PCI Express设备功能的[0型配置空间报头](#_bookmark202)的寄存器字段的

字节偏移

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

设备ID

地位

BIST

类代码

报头类型

供应商ID命令

修订ID

等待定时器

高速缓存行大小

基址寄存器

Cardbus CIS指针

子系统ID子系统供应商ID扩展ROM基地址

能力指针

保留

最大纬度

最小值\_Gnt

中断引脚

中断线

+000小时

+004小时

+008小时

+00Ch

+010小时

+014小时

+018小时

+01Ch

+020小时

+024小时

+028小时

+02Ch

+030小时

+034小时

+038小时

+03Ch

图7-10[类型0配置空间标题](#_bookmark202)

[第7.5.1.1](#_bookmark176)详细介绍了对所有配置空间标头类型有效的PCI专用寄存器本节中定义了对[类型0配置空间报头](#_bookmark202)专用的寄存器的PCI PCI专用解释

* + - * 1. *基址寄存器*（偏移10 h-24 h）

系统软件必须在引导机器到操作系统之前建立一致的地址映射这意味着它必须确定系统中有多少内存，以及系统中的函数需要多少地址空间在确定该信息之后，系统软件可以将功能映射到合理的位置并继续系统引导。为了以与设备无关的方式进行该映射，用于该映射的基址寄存器被放置在配置空间的预定义报头部分强烈建议加电固件/软件也支持可选的增强配置访问机制（ECAM）。

所有基址寄存器中的位0都是只读的，用于确定寄存器映射到内存还是I/O空间。映射到内存空间的基址寄存器必须在0位返回0 b（见[图7-11](#_bookmark204)）。映射到I/O空间的基址寄存器必须在0位返回1b（见[图7-12](#_bookmark205)）。

内存空间指示器内存类型可预取

31

基地址

4 3 2 1 0

0

图7-11存储器

保留I/O空间

31

基地址

2 1 0

0 1

图7-12 I/O

映射到I/O空间的基址寄存器始终为32位宽，位0硬连线到1b。位1是保留的，读取时必须返回0 b，其他位用于将函数映射到I/O空间。

映射到内存空间的基址寄存器可以是32位或64位宽（以支持映射到64位地址空间），其中位0硬连线到0b。对于存储器基址寄存器，位2和位1的编码含义如[表7-8](#_bookmark206)所示。如果数据可预取，则位3应设置为1b，否则设置为0b如果对读取没有副作用，则允许函数将范围标记为可预取，函数返回读取时的所有字节而不管字节启用，并且主桥可以将处理器写入合并到该范围139中而不引起错误。位3-0是只读的。

表7-8存储器基址寄存器位2：1编码

|  |  |
| --- | --- |
| 位2：1（b） | 意义 |
| 00 | 基址寄存器是32位宽的，可以映射到32位地址位存储器空间中的任何位置 |
| 01 | 保留140 |
| 10 | 基址寄存器是64位宽的，可以映射到64位地址位存储空间中的任何位置 |
| 11 | 保留 |

函数实际实现的高位数取决于函数将响应多少地址空间可以实现32位基址寄存器，以支持从16字节到2 GB的2的幂的单个内存大小一个需要1MB内存地址空间的函数（使用32位基址寄存器）

1. 任何设备，如果其范围的行为类似于正常内存，则应将该范围标记为可预取。图形设备中的线性帧缓冲区就是应标记为可预取的范围的示例
2. 支持低于1 MB内存空间的编码在PCI本地总线规范的早期版本中得到支持系统软件应识别此编码并进行适当处理

构建地址寄存器的前12位，将其他位硬连接到0。BAR中某些位的属性受可调整大小的BAR功能（如果实现了该功能）的影响

上电软件可以通过向寄存器写入全1的值然后读回该值来该函数将在所有无关地址位中返回0，有效地指定所需的地址空间未实现的基址寄存器硬连线为零。

这种设计意味着使用的所有地址空间的大小都是二的幂，并且自然对齐。函数可以自由地消耗比所需更多的地址空间，但是对于需要少于4 KB内存空间的函数，建议将其解码为4 KB内存空间。例如，具有64字节的寄存器的函数可以被映射到存储器空间中，从而消耗高达4KB的地址空间，以最小化地址解码器中的位数。如果函数的编程模型从不访问未使用的空间，则不需要消耗比其使用更多的地址空间的函数响应该地址空间允许该函数返回不支持的访问请求，以未使用的位置为目标。将控制函数映射到I/O空间的函数每个I/O基地址寄存器或中的每个条目消耗的字节不得超过256

[增强分配能力](#_bookmark2)。对于16位I/O系统（如PC兼容机）的功能，I/O基址寄存器的高16位可以硬连线为零但是，仍然必须对I/O地址进行完整的32位解码

执行说明

调整32位基址寄存器的大小示例

在调整基址寄存器大小之前，通过[命令寄存器](#_bookmark178)禁用相应地址空间的解码（I/O或内存）软件保存基址寄存器的原始值，将一个全为1的值写入寄存器，然后读回。通过首先清除编码信息位（I/O的位1：0，存储器的位3：0），反转所有32位（逻辑非），然后递增1，可以从读取的32位值进行大小计算。得到的32位值是寄存器解码的内存/I/O范围大小请注意，如果基址寄存器用于I/O，则结果的高16位将被忽略，并且位31：16在读取时返回0在重新启用功能的[命令寄存器](#_bookmark178)中的解码之前，恢复基址寄存器中的原始值

64-位（存储器）基址寄存器可以被相同地处理，除了第二个32位寄存器被认为是第一个寄存器的扩展（即，比特63：32）。软件向两个寄存器写入一个全1的值，读回它们，并将结果组合成一个64位值。大小计算在64位值上完成

类型[0配置空间报头](#_bookmark202)具有六个DWORD位置，这些DWORD位置被分配用于从配置空间中的偏移10h开始的基址寄存器一个Type 1的DWORD空间头只有两个DWORD位置。函数可以使用任何位置来实现基址寄存器。实现的64位基址寄存器占用两个连续的DWORD位置。软件查找实现的基址寄存器必须从偏移10h开始，并继续向上通过偏移24h。一个典型的函数需要一个存储器范围用于其控制功能。某些图形函数使用两个范围，一个用于控制函数，另一个用于帧缓冲区。一个要同时将控制功能映射到内存和I/O空间的函数必须实现两个基址寄存器（一个内存和一个I/O）。该函数的驱动程序可能只使用一个空间，在这种情况下，其他空间将未使用。建议函数始终将控制函数映射到内存空间。

通过BAR请求存储空间的PCI Express功能必须设置BAR的可预取位，除非该范围包含具有读取副作用的位置或该功能不容许写入合并的位置强烈建议映射到内存空间的资源尽可能设计为可预取的。传统端点以外的PCI Express功能必须支持请求可预取内存空间的任何基址寄存器的64位寻址。BAR请求的最小内存空间地址范围为128字节。BAR中某些位的属性受可调整大小的BAR功能（如果实现了该功能）的影响

# 执行说明

关于存储空间BAR中可预取位的附加指南

具有请求大量不可预取存储器空间的存储器空间BAR的PCI Express适配器超过64 MB）可能会导致某些可伸缩平台上的空间不足，因为许多平台总共只支持1 GB或更少的不可预取内存空间。这可能会限制这些平台上支持的此类适配器的数量由于这个原因，特别鼓励请求大量内存空间的BAR设置其可预取位，因为可预取内存空间在大多数可伸缩平台上更

虽然如果存储器空间BAR的范围内没有位置具有读取副作用并且所有位置都容许写入合并，则要求存储器空间BAR使其可预取位置位，但是存在其中使可预取位置位仍将允许正确操作的系统配置，即使不满足那些条件对于那些情况，适配器在某些候选BAR中设置可预取位可能是有意义的，使得系统可以将那些BAR映射到可预取存储器空间中，以便避免不可预取存储器空间短缺。

在满足以下列举的标准的PCI Express系统上，设置候选BAR中的可预取位仍将允许正确操作，即使BAR这主要是因为PCI Express内存读取始终包含显式长度，并且PCI Express交换机从不预取或进行字节合并。一般来说，只有64位BAR是很好的候选者，因为只有传统端点被允许设置32位BAR中的可预取位，并且大多数可伸缩平台将所有32位存储器BAR映射到不可预取存储器空间中，而不管可预取位值如何。

以下是足以保证给定候选BAR的正确性的标准

* 从主机到适配器的整个路径都通过PCI Express。
* 没有传统的PCI或PCI-X设备对BAR映射的范围进行对等读取
* PCI Express主桥不进行字节合并。（这在大多数平台上都是正确的
* 任何具有读取副作用的位置都不会成为TH位置位的内存读取的目标。 看到

[第2.2.5节](#_bookmark34)。

* BAR映射的范围永远不会是推测性内存读取的目标，无论是主机发起的还是对等的。

上述标准是一个简化的集合，足以保证正确性。其他限制较少但更复杂的标准也可以保证正确性，但不在本规范的范围内

* + - * 1. *Cardbus CIS指针寄存器*（偏移28h）

该寄存器最初在[[PC卡](#_bookmark33)]中描述它的功能不适用于PCI Express。它必须是只读的，并硬连接到0000 0000h。

* + - * 1. *子系统供应商ID寄存器*/*子系统ID寄存器*（偏移2Ch/2 Eh）

子系统供应商ID和子系统ID寄存器用于唯一标识PCI Express组件所在的适配器或子系统它们为供应商提供了一种机制，即使组件上可能具有相同的PCI Express组件（因此具有相同的供应商ID和设备ID），也可以将其产品彼此区分开

除了具有基类06 h和子类00 h-04 h（00 h、01 h、02 h、03 h、04 h）或基类08 h和子类00 h-03 h（00 h、01 h、02 h、03 h）的功能外，所有功能都需要实现这些寄存器。子系统供应商ID可以从PCI SIG获得，并且用于识别适配器、主板或子系统141的供应商。子系统供应商ID（SVID）必须是PCI-SIG分配给子系统供应商的供应商ID为了与PCI-SIG程序保持一致，必须从PCI-SIG获得有效的供应商标识符以确保唯一性。

子系统ID的值由供应商指定。子系统ID值与子系统供应商ID一起构成PCI产品的唯一标识符子系统ID和设备ID值是不同的，彼此无关，软件不应假定它们之间有任何关系

这些寄存器中的值必须在功能变为“调试就绪”之前加载。没有指定如何加载这些值，但是可以在制造过程中完成或从外部逻辑加载（例如，打包选项、串行ROM等）。不能使用扩展ROM软件加载这些值，因为不能保证在所有系统的开机自检期间都运行

如果器件设计为仅在系统板上使用，则系统供应商可以在每次上电后使用系统特定软件来初始化这些寄存器

执行说明

子系统供应商ID和子系统ID

子系统供应商ID和子系统ID字段一起允许软件唯一识别PCI电路板产品。因此，供应商不应在共享一个共同子系统供应商ID的多个产品类型中重复使用子系统ID值。如果产品属于相同的系列和代，并且仅在容量或性能上有所不同，则可以在具有相同子系统供应商ID的产品上重复使用子系统ID值，但不是首选还应注意，供应商可以在一段时间内为单个产品类型使用多个唯一的子系统ID值，例如指示某些内部差异，例如组件选择。

* + - * 1. *扩展ROM基址寄存器*（偏移量30h）

某些功能，特别是那些打算在附加卡上使用的功能，需要扩展ROM的本地EPROM（有关ROM内容的定义，请参阅[PCI-固件]此寄存器用于处理此扩展ROM的基址和大小信息寄存器布局如[图7-13](#_bookmark3)所示，[表7-9](#_bookmark35)描述了寄存器中的位

1. 一家公司只需要一个供应商ID。该值可以用于配置空间的供应商ID寄存器（例如，偏移00h）或配置空间的子系统供应商ID寄存器（例如，偏移2Ch）。它用于供应商ID寄存器（例如，偏移00h），如果该公司构建了硅。它用于子系统供应商ID寄存器（例如，偏移2Ch），如果公司制造了装配。如果一家公司同时制造芯片和组件，那么两个寄存器中将使用相同的