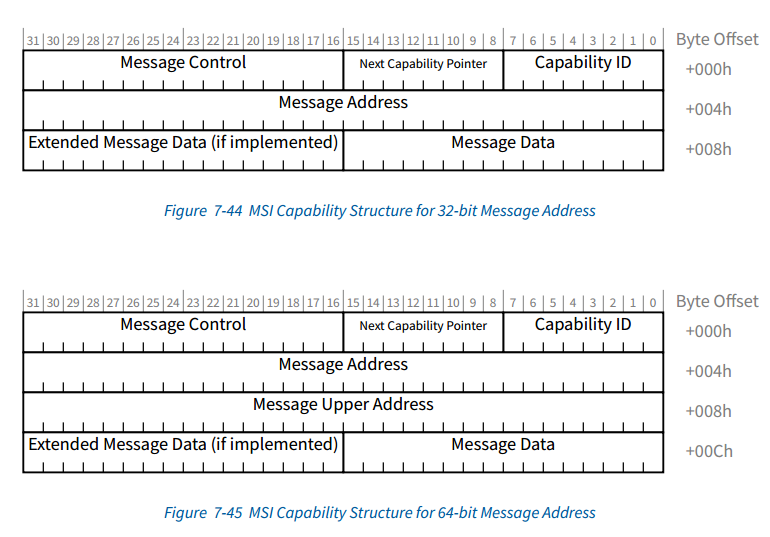
在 PCIe 设备中含有两个 Capability 结构， 一个是 MSI Capability 结构， 另一个是 MSI⁃X Capability 结构。通常情况下一个 PCIe 设备仅包含一种结构，或者为 MSI Capability 结构， 或者为 MSI⁃X Capability 结构。

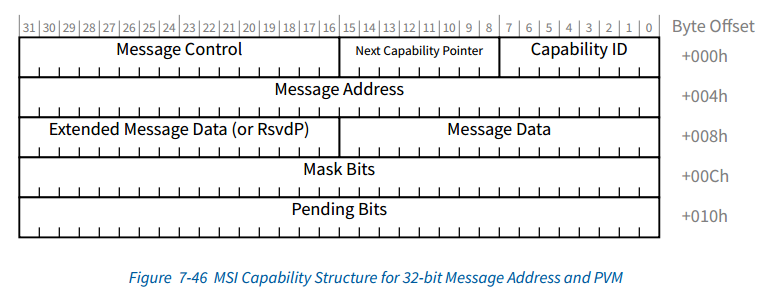
# 7.7.1 MSI Capability Structures

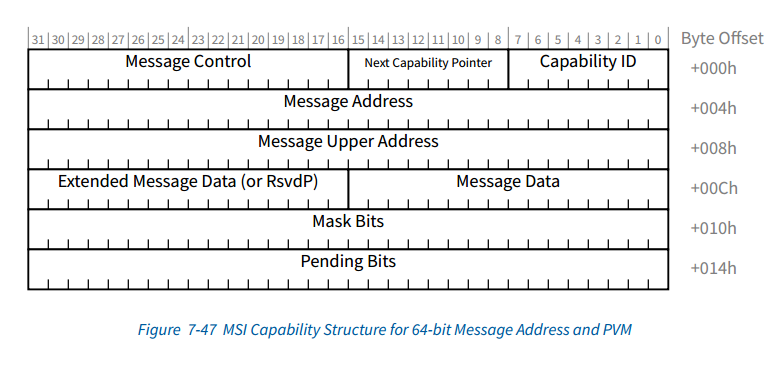
所有能够产生中断的PCI Express设备function必须实现MSI或MSI- x或两者。

本节将描述MSI Capability结构。第7.7.2节描述了MSI-X Capability结构。

MSI Capability的结构如图7-44和图7-45所示。支持MSI的每个设备功能(在多功能设备中)都必须实现自己的MSI capability结构。禁止每个function有多个MSI capability结构，但允许一个function同时拥有一个MSI和一个MSI- x capability结构。







为了请求服务，MSI PCIE设备将MSI的Message Date寄存器的内容(如果启用，则为MSI的Extended Message Data寄存器)写入MSI的Message Address寄存器的内容指定的地址(当使用64位消息地址时，可选地，MSI的Message Upper Address寄存器)。读取由Message Address寄存器的内容指定的地址会产生未定义的结果。

支持MSI的设备实现图7-44至图7-47所示的四种MSI Capability结构布局之一，具体取决于支持哪些可选功能。实现MSI的遗留端点需要支持MSI Capability结构的32位或64位消息地址版本。实现MSI的PCI Express EP需要支持MSI Capability结构的64位消息地址版本。MSI的Message Control寄存器指示设备的Capability，并提供对MSI的系统软件控制。

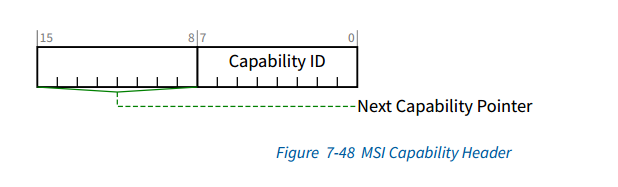
由上Capability定义可以得出：

1. MSI中断只有一个Message address同时也只有一个Message data；
2. 虽然能支持最多至32个中断向量，但是Multiple Message Capable/Enable也只是定义了个数2^N个。对于N等于3，一个 PCIe 设备需要使用 8 个中断请求， Message Data 的[2∶ 0]字段可以为 0b000 ～ 0b111， 因此可以发送 8 个中断请求， 但是这 8 个中断请求的 Message Data字段必须连续。 Message Data字段与处理器系统相关，在许多中断控制器中， Message Data 字段连续也意味着中断控制器需要为这个 PCIe 设备分配 8 个连续的中断向量号。
3. 有时在一个中断控制器中， 虽然具有 8 个以上的中断向量号， 但是很难保证这些中断向量号是连续的。 因此中断控制器将无法为这些 PCIe 设备分配足够的中断请求， 此时该设备的 “Multiple Message Enable” 字段将小于 “Multiple Message Capable”。

下面几节将进一步描述每个字段。

## 7.7.1.1 MSI Capability Header(Offset 00h)

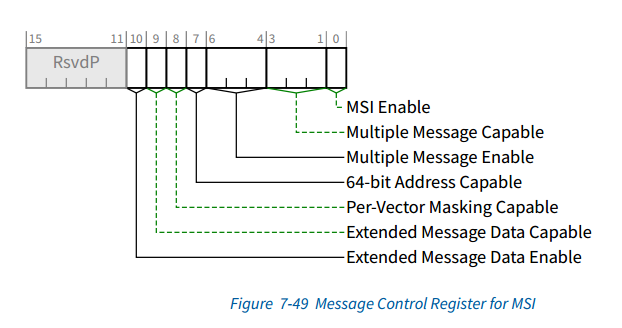
MSI Capability Header列举了PCI配置空间能力列表中的MSI Capability结构。图7-48详细描述了MSI Capability Header中寄存器字段的分配;比特定义如表7-38所示。



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 7:0 | Capability ID - MSI Capability结构。该字段必须返回一个05h的Capability ID，表示这是一个MSI Capability结构 | RO |
| 15:8 | Next Capability Pointer -该字段包含到下一个PCI Capability结构的偏移量，如果Capability链表中没有其他项，则为00h | RO |

## 7.7.1.2 Message Control Register for MSI (Offset 02h)

这个寄存器提供了对MSI的系统软件控制。缺省情况下，禁用MSI功能。如果MSI和MSI- x都被禁用，该设备使用INTx中断请求服务(如果支持)。系统软件可以通过设置该寄存器的0 bit使能MSI。允许系统软件修改MSI的消息控制寄存器的RW域段。设备驱动程序不允许修改MSI的消息控制寄存器的RW域段。



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 0 | MSI Enable -如果Set且MSI- x的消息控制寄存器(见7.9.2节)中的MSI- x Enable位为Clear，则允许该设备使用MSI请求服务，禁止使用INTx中断。系统配置软件设置此位以启用MSI。设备驱动程序禁止写入此位来掩盖设备的服务请求。关于INTx中断的控制，请参见第7.5.1.1节。  若为“清除”，则禁止该设备使用MSI请求服务。  缺省值为0b。 | RW |
| 3:1 | Multiple Message Capable -系统软件读取此字段以确定请求向量的数量。请求向量的数量必须对齐为2的幂(如果设备需要三个向量，则通过将该字段初始化为010b来请求四个)。编码定义为:  000b: 请求1个中断向量  001b: 请求2个中断向量  010b: 请求4个中断向量  011b: 请求8个中断向量  100b: 请求16个中断向量  101b: 请求32个中断向量  110b: reserved  111b: reserved | RO |
| 6:4 | Multiple Message Enable -软件写入此字段以指示分配向量的数量(等于或小于请求向量的数量)。分配的向量的数量以2的幂排列。如果一个设备请求四个向量(由多消息编码010b表示)，系统软件可以通过分别向该字段写入010b、001b或000b来分配四个、两个或一个向量。当MSI使能被设置时，一个设备将被分配至少一个向量。编码定义为:  000b: 请求1个中断向量  001b: 请求2个中断向量  010b: 请求4个中断向量  011b: 请求8个中断向量  100b: 请求16个中断向量  101b: 请求32个中断向量  110b: reserved  111b: reserved | RW |
| 7 | 64bit Address Capable -设置后，该设备可以发送64位的消息地址。清除表示该设备不能发送64位的消息地址。如果该设备是PCI Express Endpoint，则必须设置此位。 | RO |
| 8 | Per-Vector Masking Capable -如果设置，该设备支持MSI逐向量屏蔽。如果是Clear，则该设备不支持MSI逐向量屏蔽。如果该设备在SR-IOV设备中是PF或VF，则必须设置此位。 | RO |
| 9 | Extended Message Data Capable -如果设置，则该设备能够提供扩展消息数据。  “Clear”表示不支持提供扩展消息数据。 | RO |
| 10 | Extended Message Data Enable -设置后，启用扩展消息数据功能。  “清除”表示不启用“扩展消息数据”功能。  缺省值为0b。  如果Extended Message Data Capable位为1b，则该位必须为读写位;否则它必须硬连接到0b。 | RW/RO |

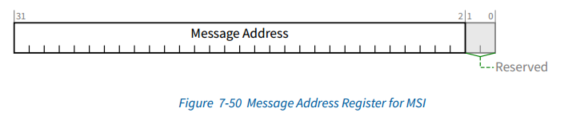
Multiple Message Capable字段都为RO寄存器，这个指的是当前PCIE设备支持的最大数量。

Multiple Message Enable字段都为RW寄存器，这个寄存器由系统软件设置，根据当前中断资源的紧张程度、策略等进行配置，Enable字段的值可能小于Capable字段的值。

Extended Message Data Capable字段表示PCIE设备提供扩展消息数据的能力。

Extended Message Data Enable字段，如果Capable为0，那么系统软件设置enable必须设置为0；如果Capable为1，那么系统软件设置enable需要根据当前使用场景去设置。

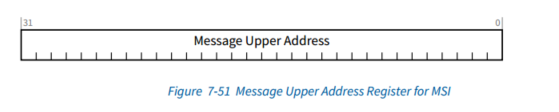
## 7.7.1.3 Message Address Register for MSI (Offset 04h)



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 1:0 | Reserved -读时总是返回0。写操作无效。 | RsvdP |
| 31:2 | Message Address -系统指定的消息地址。  如果设置了MSI使能位，则该寄存器的内容指定了MSI事务的dword对齐地址(address[31:02])。地址[1:0]设置为00b。  该字段的缺省值没有定义。 | RW |

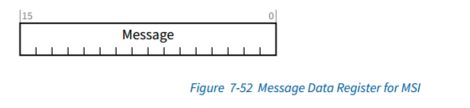
当 MSI Enable 位有效时， 该字段存放 MSI 存储器Memory write的目的地址的低 32 位。 该字段的 31∶ 2 字段有效， 系统软件可以对该字段进行读写操作； 该字段的第 1 ～ 0 位为 0。

## 7.7.1.4 Message Upper Address Register for MSI (Offset 08h)



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 31:0 | Message Upper Address -系统指定的消息上地址。  该寄存器仅在设备支持64位消息地址(设置64位地址能力)时实现。这个寄存器对于PCI Express EP是必需的，对于其他设备类型是可选的。  如果设置了MSI使能位，则该寄存器的内容(如果非零)指定64位消息地址(address[63:32])的上32位。如果该寄存器的内容为零，则该设备使用消息地址寄存器指定的32位地址。  该字段的缺省值没有定义。 | RW |

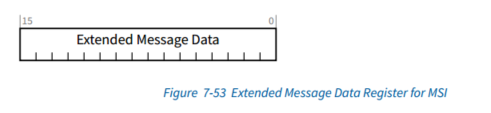
## 7.7.1.5 Message Data Register for MSI (Offset 08h or 0Ch)



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 15:0 | Message Data-系统指定的消息数据。  如果设置了MSI使能位，则该设备使用低16位的消息数据发送DWORD内存写事务。设置所有4字节启用。  Multiple Message Enable字段定义了低顺序消息Message Data位的数量，该设备被允许修改以生成其系统软件分配向量。例如，编码为010b的Multiple Message Enable表示该设备已经分配了四个中断向量，并且允许修改Meaage Data位1和0(PCIE设备修改较低的Message Data位以分配中断向量)。如果multimessage Enable字段为000b，则不允许该设备修改消息数据。  该字段的缺省值没有定义。 | RW |

当 MSI Enable 位有效时， 该字段存放 MSI 报文使用的数据。 该字段保存的数值与处理器系统相关， 在 PCIe 设备进行初始化时， 处理器将初始化该字段， 而且不同的处理器填写该字段的规则并不相同。 如果 MultipleMessage Enable 字段不为 0b000 时 （即该设备支持多个中断请求时）， PCIe 设备可以通过改变 Message Data 字段的低位数据发送不同的中断请求。如上所述，4个中断向量可以改变Message Data的低2位来生成与此对应的4种不同中断向量的Message Data值。

## 7.7.1.6 Extended Message Data Register for MSI (Optional)



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 15:0 | Extended Message Data-系统指定的消息数据。  这个寄存器是可选的。对于没有逐向量掩蔽的MSI能力结构，如果设置了扩展消息数据能力位，则必须实现该功能;否则，它在MSI Capability结构之外并且未定义。对于具有逐向量掩蔽的MSI能力结构，如果设置了扩展消息数据能力位，则必须实现该功能;否则为RsvdP。  如果设置了扩展消息数据使能位，则DWORD memory write报文使用上16位的扩展消息数据;否则，它使用0000h作为上16位。  该字段的缺省值是0000h。 | RW/undefined/  RsvdP |

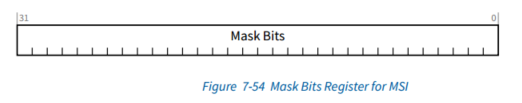
## 7.7.1.7 Mask Bits Register for MSI (Offset 0Ch or 10h)

这个寄存器是可选的。如果设置了Per-Vector Masking Capable(参见第7.7.1.2节)，则存在。这个寄存器在Capability的偏移量取决于64-bit Address Capable的值(参见章节7.7.1.2)。

掩码位和待处理位寄存器使软件能够在每个向量的基础上禁用或延迟消息发送。

MSI向量从0到N-1编号，其中N为软件分配的向量个数。每个向量在掩码位和待处理位寄存器中与相应的编号位相关联。

Multiple Message Capable字段表示实现了多少个向量(带有相关的掩码和Pending位)。所有未实现的掩码和挂起位都是保留的。



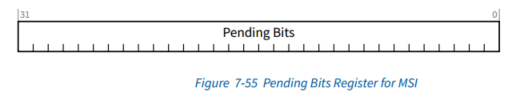
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 31:0 | Mask Bits—对于设置的每一个掩码位，禁止发送关联的报文。  默认为0。 | RW |

PCIe 总线规定当一个设备使用 MSI 中断机制时， 最多可以使用 32 个中断向量， 从而一个设备最多可以发送 32 种中断请求。 Mask Bits 字段由 32 位组成，其中每一位对应一种中断请求。 当相应位为 1 时表示对应的中断请求被屏蔽， 为 0 时表示允许该中断请求。 系统软件可读写该字段， 系统初始化时该字段为全 0， 表示允许所有中断请求。 该字段和 Pending Bits 字段对于 MSI 中断机制是可选字段， 但是PCIe 总线规范强烈建议所有 PCIe 设备支持这两个字段。

## 7.7.1.8 Pending Bits Register for MSI (Offset 10h or 14h)

这个寄存器是可选的。如果设置了Per-Vector Masking Capable(参见第7.7.1.2节)，则存在。

这个寄存器在Capability的偏移量取决于64-bit Address Capable的值(参见章节7.7.1.2)。有关该寄存器的附加要求，见7.7.1.7节。



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 31:0 | Pending Bits-对于设置的每个pending位，设备都有一个对应值来填写pending位。  默认为0。 | RO |

该字段对于系统软件是只读位， PCIe 设备内部逻辑可以改变该字段的值。 该字段由 32 位组成， 并与 PCIe 设备使用的 MSI 中断一一对应。 该字段需要与 Mask Bits 字段联合使用。当 Mask Bits 字段的相应位为 1 时， 如果 PCIe 设备需要发送对应的中断请求， PendingBits 字段的对应位将被 PCIe 设备的内部逻辑置 1， 此时 PCIe 设备并不会使用 MSI 报文向中断控制器提交中断请求； 当系统软件将 Mask Bits 字段的相应位从 1 改写为 0 时， PCIe 设备将发送 MSI 报文向处理器提交中断请求， 同时将 Pending Bit 字段的对应位清零。 在设备驱动程序的开发中， 有时需要联合使用 Mask Bits 和 Pending Bits 字段防止处理器丢弃中断请求。

# 7.7.2 MSI-X Capability and Table Structure

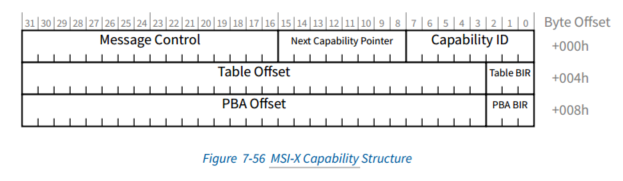
MSI-X Capability的结构如图7-56所示。禁止每个设备有一个以上的MSI- x能力结构，但允许一个设备同时具有MSI Capability结构和MSI- x Capability结构。

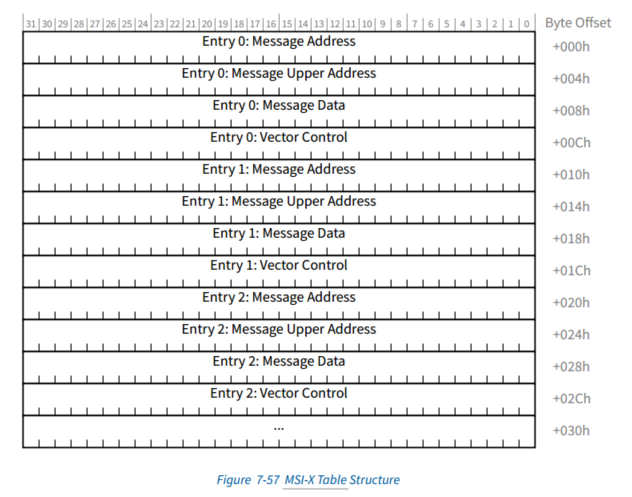
与直接包含中断向量的所有控制/状态信息的MSI Capability结构相反，MSI- x Capability结构指向一个MSI- x表结构和一个MSI- x PBA结构(Pending Bit Array结构)，它们都驻留在内存空间中(见图7-57和图7-58)。

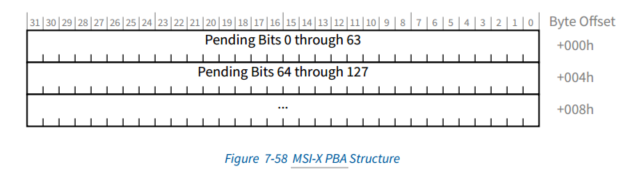
每个结构由一个属于设备的基址寄存器(BAR)映射，从配置空间的10h开始，或者在Enhanced Allocation capability中进入。BAR指示器寄存器(BIR)指示哪个BAR(或在使用Enhanced Allocation capability时使用BEI)， qword对齐偏移量指示以下结构相对于与BAR关联的基址开始的位置。BAR可以是32位或64位，但必须映射内存空间。允许一个设备用相同的BAR映射两个结构，或者用不同的BAR映射每个结构。

MSI-X表结构如图7-57所示，通常包含多个表项，每个表项由几个字段组成:Message Address、Message Upper Address、Message Data和Vector Control。每个条目都能够指定一个唯一的中断向量。

Pending Bit Array (PBA)结构，如图7-58所示，包含了设备的Pending Bits，每个表项一个，在QWORDs中被组织成一个打包的Bits数组。最后一个QWORD不一定是完全填充的。







使用给定的MSI-X table entry请求服务，设备将执行DWORD Memory Write事务，使用数据的Message Data字段项的内容， Message Upper Address字段的内容作为地址的高32位，Message Address字段项的内容作为地址的低32位。对前述指定的地址发起Memory Read事务将产生未定义的结果。

如果为MSI-X Table或MSI-X PBA映射地址空间的BAR或Enhanced Allocation capability中的条目同时也映射与MSI-X结构无关的其他可用地址空间，则在其他地址空间中使用的位置(例如，用于csr)不得与MSI-X结构占用的任何自然对齐的4KB地址空间共享。

这允许系统软件在适用的情况下为MSI-X结构和其他地址空间使用不同的处理器属性。(有些处理器体系结构不支持将不同的处理器属性与相同的自然对齐的4KB物理地址范围相关联。)允许MSI-X Table和MSI-X PBA共存于一个自然对齐的4kb地址范围内，但它们不能相互重叠。

* 专用BAR和地址范围隔离

1. 为了使系统软件能够将MSI-X结构映射到不同的处理器页面以改进访问控制，建议设备为MSI-X Table和MSI-X PBA专用单独的BAR或者提供比地址范围所需的最低隔离更多的隔离。
2. 如果专用的单独的BAR是不可行的，建议一个设备为MSI-X Table和MSI-X PBA设立一个专用的基址寄存器。
3. 如果一个专用的基址寄存器是不可行的，建议一个设备隔离MSI-X结构与非MSI-X结构用8 KB对齐，而不是强制用4 KB对齐。

* 例如，如果一个Base Address Register需要为包含128个entry的MSI-X Table映射2 KB，为包含128位的MSI-X PBA映射16字节，为与MSI-X无关的寄存器映射64字节，那么以下是一个可接受的实现。BAR请求8kb的总地址空间，映射非MSI-X寄存器的前64字节，映射以4kb偏移开始的MSI-X表，映射以6kb偏移开始的MSI-X PBA。
* 对于共享基地址寄存器来说，一个较好的实现是请求16 KB的总地址空间，映射非MSI-X寄存器的前64字节，映射以8 KB偏移开始的MSI-X Table，映射以12 KB偏移开始的MSI-X PBA。
* 读写内存中的MSI-X内存空间结构

MSI-X Table和MSI-X PBA结构的定义使得它们可以存在设备上的通用读/写内存中，以便于实现和增加灵活性。要实现这一点，所包含的字段都不需要是只读的，并且对事务对齐和大小也有限制。

对于所有访问MSI-X表和MSI-X PBA字段，软件必须使用对齐的全DWORD或对齐的全QWORD事务;否则，结果为未定义。

MSI-X Table entries和Pending bits分别从0到N-1编号，其中N-1由MSI-X的消息控制寄存器中的表大小字段控制。对于任意给定的MSI-X Table项k，其起始地址可以用公式计算:

entry starting address = Table base + k × 16

对于关联的Pending位k，其QWORD访问地址和该QWORD内的位数可以用公式计算:

QWORD address = PBA base + (k div 64) × 8

QWORD bit = k mod 64

选择通过DWORD访问读取pending bit K的软件可以使用这些公式:

DWORD address = PBA base + (k div 32) × 4

DWORD bit = k mod 32

MSI-X Capability、MSI-X Table和MSI-X PBA结构中的每个字段将在以下章节中进一步描述。

在MSI-X Capability结构中，读时保留寄存器和位总是返回0，而写操作没有影响。在MSI-X表和PBA结构中，保留字段有特殊的规则。

在 MSI⁃X Capability 结构中， 每一个中断请求都使用独立的 Message Address 字段和 Message Data 字段， 从而中断控制器可以更加合理地为该设备分配中断资源。

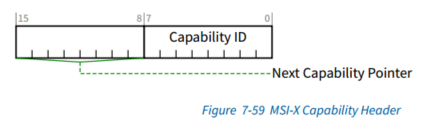
除此之外 MSI⁃X 中断机制还使用了独立的 Pending Table 表， 该表用来存放与每一个中断向量对应的 Pending 位。 这个 Pending 位的定义与 MSI Capability 寄存器的 Pending 位类似。

MSI⁃X Table 和 Pending Table 存放在 PCIe 设备的 BAR 空间中。 MSI⁃X 机制必须支持这个Pending Table， 而 MSI 机制的 Pending Bits 字段是可选的。

## 7.7.2.1 MSI-X Capability Header(Offset 00h)

MSI-X Capability Header列举了PCI Configuration Space Capability列表中的MSI-X Capability结构。

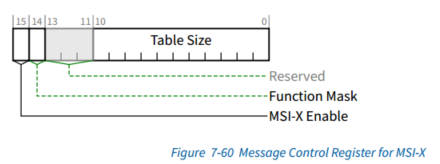
图7-56详细描述了MSI-X Capability Header中寄存器字段的分配;比特定义如表7-46所示。



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 7:0 | Capability ID - MSI Capability结构。该字段必须返回一个11h的Capability ID，表示这是一个MSI Capability结构 | RO |
| 15:8 | Next Capability Pointer -该字段包含到下一个PCI Capability结构的偏移量，如果Capability链表中没有其他项，则为00h | RO |

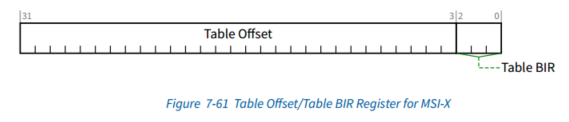
## 7.7.2.2 Message Control Register for MSI-X (Offset 02h)

缺省情况下，禁用MSI-X功能。如果MSI和MSI- x都被禁用，则该设备通过INTx中断请求服务(如果支持)。系统软件可以通过设置该寄存器的第15位使能MSI-X。系统软件允许修改消息控制寄存器的RW字段。设备驱动程序不允许修改消息控制寄存器的读写位和字段。



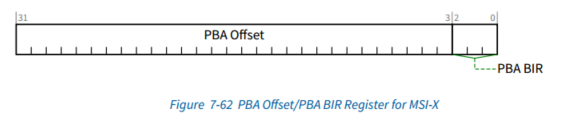
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 10:0 | Table Size-系统软件读取该字段以确定MSI-X表大小N，编码为N-1。例如，返回值000 0000 0011b表示表大小为4。 | RO |
| 13:11 | Reserved -读时总是返回0。写操作无效。 | RsvdP |
| 14 | Function Mask-如果设置，与函数相关的所有向量都被掩码，无论它们的每个向量掩码位值如何。  如果Clear，则每个向量的Mask位决定该向量是否被屏蔽。  设置或清除MSI-X功能掩码位对每向量掩码位的值没有影响。  缺省值为0b | RW |
| 15 | MSI-X Enable -如果Set和MSI的消息控制寄存器(见7.7.1.2节)中的MSI使能位为Clear，则允许该功能使用MSI- x请求服务，禁止使用INTx中断(如果实现)。系统配置软件设置此位以启用MSI-X。设备驱动程序禁止写入此位来掩盖函数的服务请求。  若为“清除”，则禁止该功能使用MSI-X请求业务。  缺省值为0b。 | RW |

## 7.7.2.3 Table Offset/Table BIR Register for MSI-X (Offset 04h)



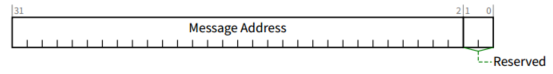
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 2:0 | Table BIR -指示一个设备BAR寄存器的哪一个，从配置空间的10h开始，或者在Enhanced Allocation capability中有一个匹配的BAR等效指示器(BEI)的条目，用于将函数的MSI-X表映射到内存空间。  定义的编码是:  0 Base Address Register 10h  1 Base Address Register 14h  2 Base Address Register 18h  3 Base Address Register 1Ch  4 Base Address Register 20h  5 Base Address Register 24h  6 Reserved  7 Reserved | RO |
| 31:3 | Table Offset - MSI-X Table在BAR空间中的偏移。较低的3表BIR位被软件屏蔽(设置为零)，以形成32位qword对齐偏移量。 | RO |

## 7.7.2.4 PBA Offset/PBA BIR Register for MSI-X (Offset 08h)



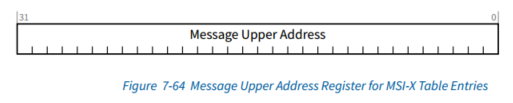
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 2:0 | PBA BIR - 指示用于将设备的MSI-X PBA映射到内存空间的BAR的哪一个(从配置空间的10h开始)或带有匹配BEI的Enhanced Allocation capability项。  PBA BIR值定义与Table BIR的值定义相同。在通常情况下，Pending Table 和 MSI⁃X Table 存放在 PCIe 设备的同一个 BAR 空间中。 | RO |
| 31:3 | PBA Offset – PBA在BAR空间的偏移。较低的3 PBA BIR位被软件屏蔽(设置为零)，以形成32位qword对齐偏移量。 | RO |

## 7.7.2.5 Message Address Register for MSI-X Table Entries



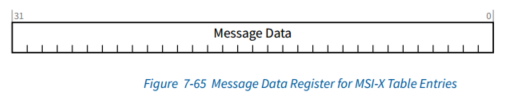
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 1:0 | 保留-为了正确的DWORD对齐，软件必须始终将这两个位写入零;否则结果是未定义的。  该字段的缺省值为00b。  这些位允许为只读或读写。 | RO或RW |
| 31:2 | 消息地址-系统指定的消息低位地址。  对于MSI-X消息，来自MSI-X表项的这个字段的内容为Memory Write事务指定了dword对齐地址的较低部分。  该字段的缺省值没有定义。 | RW |

## 7.7.2.6 Message Upper Address Register for MSI-X Table Entries



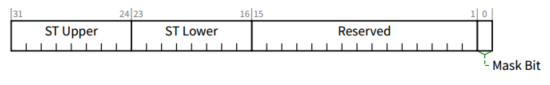
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 31:0 | Message Upper Address -系统指定的消息高地址位。  如果该字段为零，则使用32位地址消息。如果该字段不为零，则使用64位地址消息。  该字段的缺省值没有定义 | RW |

## 7.7.2.7 Message Data Register for MSI-X Table Entries



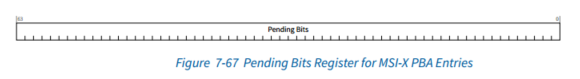
|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 31:0 | Message Data - 系统指定的消息数据。  对于MSI-X消息，来自MSI-X表项的这个字段的内容指定DWORD Memory Write事务的32位数据负载。设置所有4字节启用。  与用于MSI消息的消息数据相反，MSI- x消息中的低位消息数据位不会被设备修改。  该字段为可读写。  该字段的缺省值没有定义。 | RW |

## 7.7.2.8 Vector Control Register for MSI-X Table Entries



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 0 | Mask Bit（Per Vector Mask） - 设置此位时，禁止使用此MSI-X表项发送消息。然而，使用相同向量编程的任何其他MSI-X entries仍然能够发送等效的消息，除非它们也被屏蔽。Per Vector Mask 位的使用方法与 MSI 机制的 Mask 位类似。 | RW |
| 15:1 | Reserved - 缺省情况下，这些位的值必须为0。但是，为了将来可能的使用，软件在修改其他矢量控制位的值时必须保留这些保留位的值。如果软件修改了这些保留位的值，则结果是未定义的。  这些位被允许为RsvdP或读写。 |  |
| 23:16 | ST Lower -如果设备实现了TPH Requester Extended Capability结构，并且ST Table Location指示值为10b，则该字段包含Steering Tag的低8位，并且必须是读写的。  否则，该字段允许读写或RsvdP，并且为了将来可能的使用，软件在修改其他矢量控制位的值时必须保留这些保留位的值，否则结果是未定义的。  该字段的缺省值是00h。 | RW或RsvdP |
| 31:24 | ST Upper - 如果功能实现了TPH Requester Extended Capability结构，并且ST Table Location指示值为10b，并且设置了Extended TPH Requester Supported位，则该字段包含转向标签的高8位，必须为读写。  否则，该字段允许读写或RsvdP，并且为了将来可能的使用，软件在修改其他矢量控制位的值时必须保留这些保留位的值，否则结果是未定义的。  该字段的缺省值是00h。 | RW或RsvdP |

## 7.7.2.9 Pending Bits Register for MSI-X PBA Entries



|  |  |  |
| --- | --- | --- |
| 域段 | 描述 | 属性 |
| 63:0 | Pending Bits - 对于设置的每个Pending位，该设备有一个与MSI-X entry相关的Pending消息。  没有关联MSI-X entry的pending bits被保留。缺省情况下，Reserved Pending bits的值必须为0b。  软件不应该写，而应该只读Pending Bits。如果软件写入Pending Bits，结果是未定义的。  每个Pending Bit的默认值为0b。  这些位允许为只读或读写。 | RO或RW |

与 MSI 机制类似， Pending 位需要与 Per Vector Mask 位配置使用。

当 Per Vector Mask 位为 1 时， PCIe 设备不能立即发送 MSI⁃X 中断请求， 而是将对应的Pending 位置 1； 当系统软件将 Per Vector Mask 位清零时， PCIe 设备需要提交 MSI⁃X 中断请求， 同时将 Pending 位清零。