
PCI
SIG™

Optimum Usage of the MSI / MSI-X Interrupt Signaling Mechanisms

Joe Cowan

Computer Systems Architect

Hewlett-Packard Company

Presentation Summary

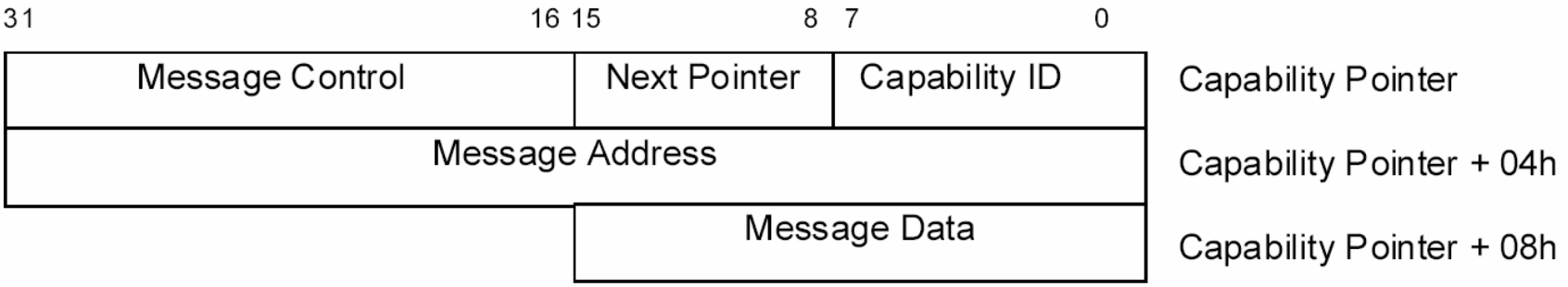
- MSI / MSI-X: Background & History
- MSI Fundamentals
  - ✓ Original Capability Structures
  - ✓ Key Attributes
  - ✓ Benefits over the INTx Mech
- MSI-X Fundamentals
  - ✓ Motivations for Extending MSI
  - ✓ MSI-X Differences from MSI
  - ✓ Capability & Other Structures
- Servicing MSI & MSI-X Interrupts
- Requirements & Special Considerations
  - ✓ Conventional PCI
  - ✓ PCI-X
  - ✓ PCI Express
- Recommendations & Guidelines
  - ✓ System Platforms
  - ✓ Adapter IHVs
  - ✓ OSs & Drivers
- Presentation Conclusions

MSI / MSI-X: Background & History

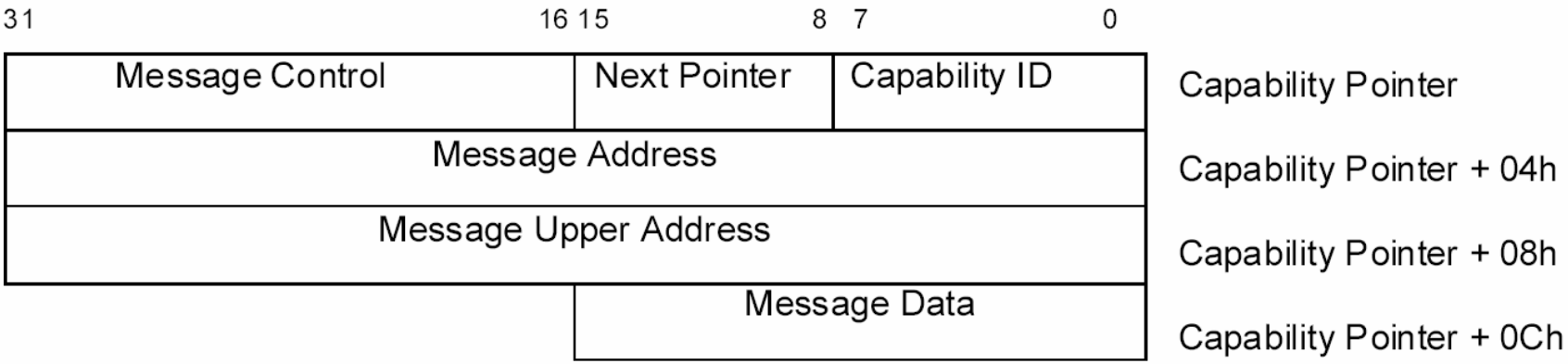
- MSI & MSI-X are interrupt signaling mechanisms
  - ✓ Interrupt is signaled via a DWORD Memory Write transaction
  - ✓ MSI & MSI-X define separate independent Capability Structures
  - ✓ Functions implement MSI, MSI-X, both, or in some cases neither
- MSI – “Message-Signaled Interrupts”
  - ✓ MSI introduced in Conventional PCI 2.2
  - ✓ MSI incorporated into PCI-X 1.0
  - ✓ MSI incorporated into PCI Express Base 1.0
- MSI-X – “MSI Extended”
  - ✓ MSI-X introduced as an ECN to PCI 2.3
  - ✓ MSI-X incorporated into PCI 3.0
  - ✓ MSI-X support added to PCI-X PT 2.0a via ECN
  - ✓ MSI-X support added to PCI Express Base 1.0a via ECN

MSI Fundamentals: Original Capability Structures

Capability Structure for 32-bit Message Address



Capability Structure for 64-bit Message Address



MSI Fundamentals: Key Attributes

- Function signals interrupt by sending DWORD Memory Write transaction using special Address & Data
  - ✓ Each unique Address/Data combination is called a “vector”
  - ✓ MSI transaction (PMW) remains ordered wrt DMA writes
  - ✓ SW must deal with event-triggered model (vs level-triggered)
  - ✓ Driver SW is prohibited from using MSI Enable bit as a mask
  - ✓ Standard PCI rules apply wrt using SAC vs DAC
- Vector management paradigm
  - ✓ Function HW *requests* 2^N vectors (up to 32)
  - ✓ FW or OS *allocates* 2^M vectors (up to # requested)
  - ✓ Function must “deal with” fewer allocated than requested
- If multiple vectors are allocated, vectors are differentiated only via LSBs in Data
  - ✓ All vectors for a Function use the same Address
  - ✓ Upper 16 bits in Data always zero

MSI Fundamentals: Benefits over the INTx Mech

- Reduction of PIO reads in performance paths
  - ✓ PIO reads are generally expensive, especially through PPBs
  - ✓ Not needed to ensure message ordering wrt DMA writes
  - ✓ Needed less often to determine interrupt source
    - Much less likely for multiple Functions to share a vector
    - Less likely for multiple sources in same Function to share a vector
- Supports multiple vectors per Function
  - ✓ Only a single INTx# line can be used per Function
  - ✓ A multi-vector Function can distribute work over multiple CPUs
    - Can be useful for load balancing
    - Can be useful for supporting multiple QoS levels
    - Can be useful for supporting processor affinity
- *But, not all platforms support multiple MSI vectors well*

MSI-X Fundamentals: Motivations for Extending MSI

- Original motivations
  - ✓ Support a higher maximum number of vectors per Function
    - MSI limited to 32 vectors
    - MSI-X supports up to 2048 vectors
  - ✓ Reduce required chipset interrupt re-vectoring logic
    - MSI uses identical address for all vectors in a Function
    - MSI differentiates vectors in a Function only by LSBs in data
    - MSI-X supports independent address/data for each vector, avoiding the need for interrupt re-vectoring logic on some platforms
- Additional motivations
  - ✓ Lack of an architected MSI masking mechanism
    - MSI not controlled by the (INTx) interrupt disable bit
  - ✓ Lack of an architected MSI polling mechanism
    - MSI status not indicated by the (INTx) interrupt status bit

MSI-X Fundamentals: MSI-X Differences from MSI

- *Note: MSI-X supports a proper superset of MSI's functionality*
  - ✓ MSI-X can generate any Memory Write transaction that MSI can
- Fundamental differences
  - ✓ New and distinct MSI-X Capability structure
  - ✓ Not backwards compatible with MSI SW
  - ✓ Number of vectors specified with unit granularity, not power-of-2
  - ✓ Most per-vector info stored in MSI-X Table; resides in Memory space
  - ✓ Bit-per-vector Pending Bit Array (PBA); resides in Memory space
  - ✓ Memory space for MSI-X Table & PBA allocated via 1 or 2 BARs
  - ✓ MSI-X Table & PBA structs architected for placement in RW memory
- Additional features
  - ✓ Mandatory per-vector masking (PVM) and Function-masking features
    - Architected masking mechanisms
    - Architected polling mechanism
    - Function-masking feature masks entire Function with single PIO write
  - ✓ PVM also defined as new optional feature for MSI
    - Two additional MSI Capability structs defined to support PVM
- Better vector management paradigm (*however, is OS dependent*)
  - ✓ Driver SW can determine how many vectors to request
  - ✓ Driver SW can manage vector assignment to specific Function int sources
  - ✓ Driver SW can manage vector sharing if/when required

MSI-X Fundamentals: Capability & Other Structures

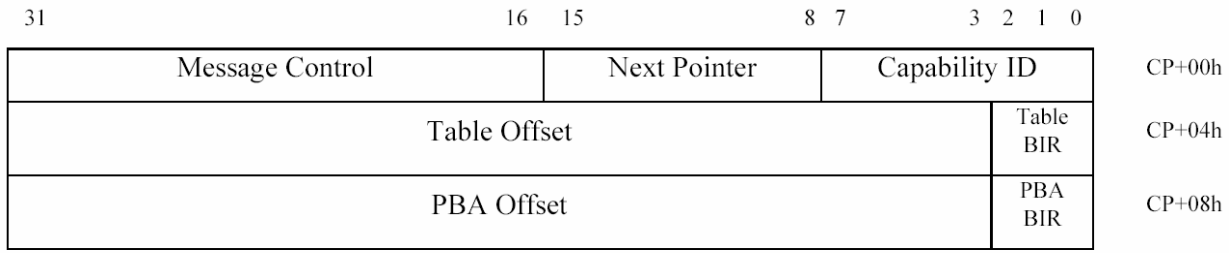


Figure 6-2: MSI-X Capability Structure

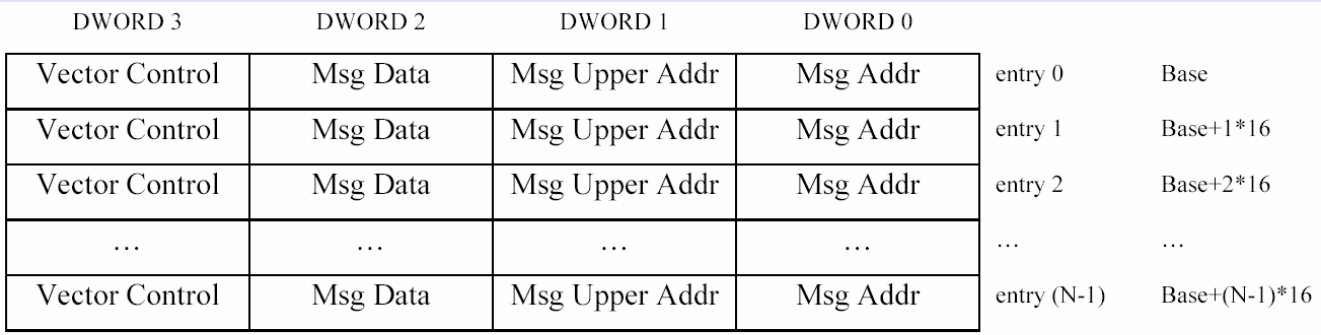


Figure 6-3: MSI-X Table Structure

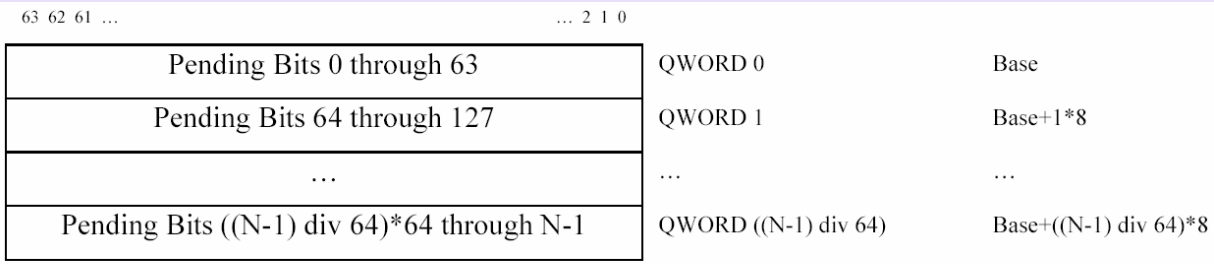


Figure 6-4: MSI-X PBA Structure

Servicing MSI & MSI-X Interrupts

- Must have proper handshakes between HW & SW, else…
  - ✓ HW may send fewer messages than SW expects (“lost/dropped interrupts”)
  - ✓ HW may send more messages than SW expects (“spurious interrupts”)
- *Question: how does HW know when to send new messages?*
  - ✓ Following are a few representative approaches
- Event queue for a vector (sophisticated but HW intensive)
  - ✓ HW/SW each manage a producer or consumer pointer, based on role
  - ✓ When SW ISR thinks it’s done, writes updated producer/consumer pointer to HW
  - ✓ If/when HW knows of more work for SW to do, generate new message
- Multiple interrupt sources for a vector (where HW supports PVM)
  - ✓ HW sends message based on assertion edge of ORed source signals
  - ✓ SW ISR upon entry uses PVM to mask further messages
  - ✓ SW services any interrupt events it discovers, then clears mask
  - ✓ If any events still pending when mask is cleared, HW sends new message
- Multiple interrupt sources for a vector (where HW does not support PVM)
  - ✓ HW sends message based on assertion edge of ORed source signals
  - ✓ After servicing presumed “last” event, SW re-inspects *all* sources
  - ✓ Re-inspection typically via PIO reads, and may cost significant CPU overhead
  - ✓ Spurious interrupts will generally occur with this scheme

Conventional PCI: Reqs & Special Considerations

- Context: for Functions that signal interrupts
- Notable MSI requirements – PCI 2.2
  - ✓ MSI is optional; if implemented…
    - 64b Addr version is required if Function supports DAC as a master
    - HW is forbidden to use INTx# if/when MSI is enabled
- Notable requirements – MSI-X ECN for PCI 2.2
  - ✓ MSI-X is optional; if implemented…
    - SW is forbidden to enable MSI & MSI-X at same time
- Special Considerations
  - ✓ INTx# pin is still recommended
  - ✓ If MSI is implemented, PVM version is recommended
  - ✓ A Function is permitted to implement both MSI & MSI-X
    - Only makes sense if backward compatibility w/ MSI SW is req'ed

PCI-X: Reqs & Special Considerations

- Context: for Functions that signal interrupts
- Notable MSI requirements – PCI-X 1.0
  - ✓ MSI is mandatory
  - ✓ 64b Address version is mandatory
- Notable requirements – MSI-X ECN for PCI-X PT 2.0a
  - ✓ Required to implement MSI or MSI-X or both
    - – If MSI-X is implemented, not required to implement MSI
- Special Considerations
  - ✓ INTx# pin is still recommended
  - ✓ If MSI is implemented, PVM version is recommended
  - ✓ A Function is permitted to implement both MSI & MSI-X
    - – Only makes sense if backward compatibility w/ MSI SW req'ed

PCI Express: Reqs & Special Considerations

- Context: for Functions that signal interrupts
- Notable MSI requirements – PCI Express 1.0
  - ✓ MSI is mandatory
  - ✓ Legacy Endpoints – either 32b or 64b Address is acceptable
  - ✓ Express Endpoints – 64b Address is required
  - ✓ RC Integrated Endpoints – either 32b or 64b Addr is acceptable
- Notable reqs – MSI-X ECN for PCI Express Base 1.0a
  - ✓ Required to implement MSI or MSI-X or both
    - – If MSI-X is implemented, not required to implement MSI
- Special Considerations
  - ✓ INTx virtual wire interrupt mechanism is still recommended
  - ✓ If MSI is implemented, PVM version is recommended
  - ✓ A Function is permitted to implement both MSI & MSI-X
    - – Only makes sense if backward compatibility w/ MSI SW req'ed
  - ✓ MSI or MSI-X messages can use TCs other than 0

System Platforms: Recommendations & Guidelines

- Support the MSI / MSI-X HW signaling mechanisms ASAP
- Continue to support the INTx mechanism

Adapter IHVs: Recommendations & Guidelines

- Context: for Functions that signal interrupts
- Implement MSI if only a single vector is required
  - ✓ Implement the 64b Address version
  - ✓ Implement PVM
- Implement MSI-X if multiple vectors are required
  - ✓ If backwards compatibility with MSI SW is required (anticipated to be rare), also implement MSI
- Continue to implement the INTx mechanism

OSs & Drivers: Recommendations & Guidelines

- Support MSI / MSI-X ASAP
- Implement the new vector management paradigm for MSI-X
  - ✓ Driver SW determines how many vectors to request
  - ✓ Driver SW specifies vector attributes when requesting vectors, and can determine the attributes of vectors that are allocated
  - ✓ Example attributes are CPU assignment & priority
  - ✓ Driver SW manages vector assignment to specific Function interrupt sources
  - ✓ Driver SW manages vector sharing if/when required
- Continue to support INTx

Presentation Conclusions

- MSI and MSI-X are strongly preferred over the INTx mechanism, especially for PCI-X & Express
- However, it's still best to implement the INTx mechanism as well in the near term
- Requirements for MSI / MSI-X vary between Conventional PCI, PCI-X, and PCI Express
- MSI-X strongly recommended for multi-vector Functions
- MSI w/ PVM recommended for single-vector Functions
- Well-designed HW/SW handshake schemes are needed to handle MSI / MSI-X event-triggered interrupts properly
- New vector management paradigm is needed for MSI-X

Thank you for attending the
PCI-SIG Developers Conference 2004.

For more information please go to
www.pcisig.com


PCI
SIG™