PCI-SIG ENGINEERING CHANGE NOTICE

TITLE:	Enhanced Allocation (EA) for Memory and I/O Resources
DATE:	Introduced: 19 March 2014 Updated: 23 October 2014 Final Approval: 23 October 2014
AFFECTED DOCUMENT:	PCI Local Bus Specification Revision 3.0, 3.1 PCI Code and ID Assignment Specification
SPONSOR:	Dave Harriman (Intel)

Part I

1. Summary of the Functional Changes

Enhanced Allocation is an optional Conventional PCI Capability that may be implemented by Functions to indicate fixed (non reprogrammable) I/O and memory ranges assigned to the Function, as well as supporting new resource "type" definitions and future extensibility to also support reprogrammable allocations.

2. Benefits as a Result of the Changes

The goals of the Enhanced Allocation Capability include:

1. Minimize implementation and validation costs, by –
    a. Simplifying integration of non-PCI IP that (almost universally) expects fixed address assignment for MMIO
    b. Simplifying validation for PCI-based SoC by eliminating corner cases related to reprogramming that, in practice, doesn't/shouldn't occur in embedded environments
    c. Reducing system bring-up times by eliminating the disconnects that can occur when hardware is not self-describing (as when firmware discovery mechanisms are used in place of PCI HW-based mechanisms)
    d. Simplifying resource allocation for VF's
2. Enable the revision and extension of the legacy prefetchable and non-prefetchable memory properties
3. Support larger numbers of resource ranges than possible with the current BAR/Base+Limit mechanisms
4. Minimize incompatibilities with older operating systems to the extent that they should be able to boot without crashing or hanging, although Functions depending on Enhanced Allocation may not be capable of functioning normally without the use of new system software

The reasons why these goals are not satisfied by existing mechanisms such as ACPI include:

1. Maintaining consistency between hardware and firmware adds cost to platform development.

2. Self-describing hardware reduces the likelihood of bugs due to inconsistencies, e.g. between hardware implementation and BIOS description.
3. Firmware typically varies from platform to platform, so placing the hardware description in the hardware itself reduces overall costs by eliminating the need to update numerous different firmware implementations (which can often number in the 10's to 100's) associated with a given hardware platform.
4. Modular SoC architectures are often built in many different variations, in some cases with board built-time options modifying which functions are enabled. The resource requirements for specific blocks will vary from one variation to another, making it difficult to maintain resource locations in "human friendly" blocks. By tying the resources specifically to functions that are enabled in a given variation, this ECN ensures that the resource descriptions track with the enabling/disabling of each individual block

3. Assessment of the Impact

As noted in (2), most Functions implementing Enhanced Allocation will not work with existing OS software. The intended use of Enhanced Allocation is for Functions in systems where both the hardware and software environments are known, and where constraints on field modifications to the system ensure that incompatible HW/SW cannot be added.

4. Analysis of the Hardware Implications

Hardware changes are required to take advantage of this new Optional capability.

5. Analysis of the Software Implications

As noted in (2) and (3) above, hardware implementing this new Optional capability will typically not be operable with existing SW. As is described in the text, HW implementing this Optional capability must be clearly documented to avoid confusion and ensure required SW constraints are satisfied.

6. Analysis of the C&I Test Implications

It is expected that HW implementing this Optional capability will pass existing C & I tests. New C & I tests would be required in order to evaluate the implementation of this capability.

Part II

Detailed Description of the change

In the Conventional PCI Specification, add/edit as shown:

3.2.2. Addressing

PCI targets (except host bus bridges and functions that rely on the Enhanced Allocation capability) are required to implement Base Address register(s) to request a range of addresses which can be used to provide access to internal registers or functions (refer to Chapter 6 for more details). The configuration software uses the Base Address register to determine how much space a device requires in a given address space and then assigns (if possible) where in that space the device will reside.

IMPLEMENTATION NOTE

Device Address Space

It is highly recommended, that a device request (via Base Address register(s) or Enhanced Allocation capability) that its internal registers be mapped into Memory Space and not I/O Space.

…

When a transaction is initiated on the interface, each potential target compares the address with its Base Address register(s) or Enhanced Allocation capability to determine if it is the target of the current transaction.

...

3.2.2.1. I/O Space Decoding

...

I/O Space enable bit is set (i.e., without the use of Base Address Registers or Enhanced Allocation capability) is referred to as a legacy I/O device.

...

3.2.3. Byte Lane and Byte Enable Usage

…

If a target supports prefetching ~~(bit 3 is set in the Memory Base Address register -- refer to Section 6.2.5.1)~~, it must also return all data

…

6. Configuration Space

...

6.2.5.1. Address Maps

After determining this information, power-up software can map the I/O controllers into reasonable locations and proceed with system boot. In order to do this mapping in a device independent manner, the base registers for this mapping are placed in the predefined header portion of Configuration Space. It is strongly recommended that power-up firmware/software also support the optional Enhanced Allocation mechanism (see Section 6.9).

…

Devices that map control functions into I/O Space must not consume more than 256 bytes per I/O Base Address register or per each entry in the Enhanced Allocation capability.

...

6.2.5.2. Expansion ROM Base Address Register

In order to minimize the number of address decoders needed, a device may share a decoder between the Expansion ROM Base Address register and other Base Address registers or entry in the Enhanced Allocation capability.[47] When expansion ROM decode is enabled, the decoder is used for accesses to the expansion ROM and device independent software must not access the device through any other Base Address registers or entry in the Enhanced Allocation capability.

…

[47]Note that it is the address decoder that is shared, not the registers themselves. The Expansion ROM Base Address register and other Base Address registers or entries in the Enhanced Allocation capability must be able to hold unique values at the same time.

...

6.8.2. MSI-X Capability and Table Structures

…

Each structure is mapped by a Base Address register (BAR) belonging to the function, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability. A BAR Indicator register (BIR) indicates which BAR (or BEI when using Enhanced Allocation), and a QWORD-aligned Offset indicates where the structure begins relative to the base address associated with the BAR. The BAR is permitted to be either 32-bit or 64-bit, but must map Memory Space. A function is permitted to map both structures with the same BAR, or to map each structure with a different BAR.

…

If a Base Address register or entry in the Enhanced Allocation capability that maps address space for the MSI-X Table or MSI-X PBA also maps other usable address space that is not associated with MSI-X structures, locations (e.g., for CSRs) used in the other address space must not share any naturally aligned 4-KB address range with one where either MSI-X structure resides.

…

6.8.2.4. Table Offset/Table BIR for MSI-X

…
Indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BEI, is used to map the function's MSI-X Table into Memory Space.
…

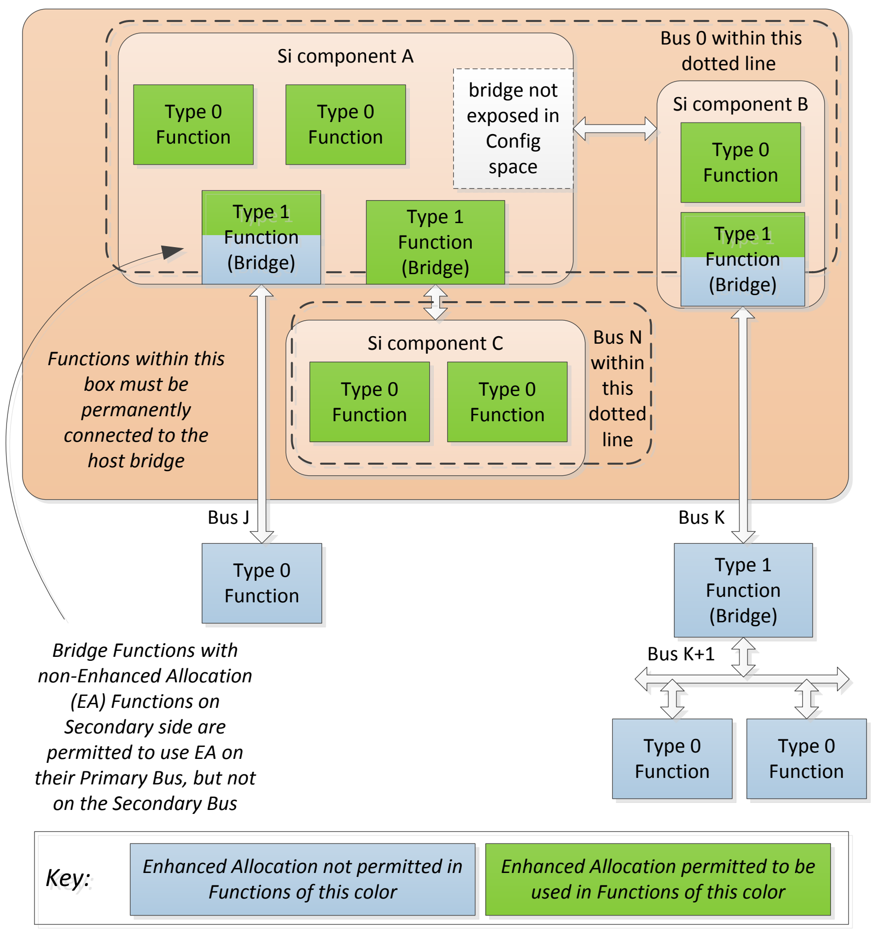
6.8.2.5. PBA Offset/PBA BIR for MSI-X

Indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, or entry in the Enhanced Allocation capability with a matching BEI, is used to map the function's MSI-X PBA into Memory Space.
…

6.9. Enhanced Allocation (EA)

The Enhanced Allocation (EA) Capability is an optional Capability that allows the allocation of I/O, Memory and Bus Number resources in ways not possible with the BAR and Base/Limit mechanisms in the Type 0 and Type 1 Configuration Headers.

It is only permitted to apply EA to certain functions, based on the hierarchal structure of the functions as seen in PCI configuration space, and based on certain aspects of how functions exist within a platform environment (see **Figure 6-1**).



Si component A
Bus 0 within this dotted line
bridge not exposed in Config space
Si component B
Type 0 Function
Type 0 Function
Type 0 Function
Type 1 Function (Bridge)
Type 1 Function (Bridge)
Type 1 Function (Bridge)
Si component C
Bus N within this dotted line
Type 0 Function
Type 0 Function
Functions within this box must be permanently connected to the host bridge
Bus J
Bus K
Type 0 Function
Type 1 Function (Bridge)
Bus K+1
Type 0 Function
Type 0 Function
Bridge Functions with non-Enhanced Allocation (EA) Functions on Secondary side are permitted to use EA on their Primary Bus, but not on the Secondary Bus
Key:
Enhanced Allocation not permitted in Functions of this color
Enhanced Allocation permitted to be used in Functions of this color

Figure 6-1: Example Illustrating Application of Enhanced Allocation

Only functions that are permanently connected to the host bridge are permitted to use EA. A bridge function (i.e., any function with a Type 1 Configuration Header), is permitted to use EA for both its Primary Side and Secondary Side if and only if the function(s) behind the bridge are also permanently connected (below one or more bridges) to the host bridge, as shown for "Si component C" in **Figure 6-1**.

A bridge function is permitted to use EA only for its Primary Side if the function(s) behind the bridge are not permanently connected to the bridge, as with the bridges above Bus J and Bus K in **Figure 6-1**, and in this case the non-EA resource allocation mechanisms in the Type 1 Header for Bus numbers, MMIO ranges and I/O ranges are used for the Secondary side of the bridge. System software must ensure that the allocated Bus numbers are within the range indicated in the Fixed Secondary Bus Number and Fixed Subordinate Bus Number registers of the EA capability. System software must ensure that the allocated MMIO and I/O ranges are within ranges indicated with the corresponding Properties in the EA capability for resources to be allocated behind the bridge. For Bus numbers, MMIO and I/O ranges behind the bridge, hardware is permitted to indicate overlapping ranges in multiple bridge functions, however, in such cases, system software must ensure that the ranges actually assigned are non-overlapping.

Functions that rely exclusively on EA for I/O and Memory address allocation must hardwire all bits of all BARs in the PCI Header to 0. Such Functions must be clearly documented as relying on EA for correct operation, and platform integrators must ensure that only EA-aware firmware/software are used with such Functions.

When a Function allocates resources using EA and indicates that a resource range is associated with an equivalent BAR number, the Function must not request resources through the equivalent BAR, which must be indicated by hardwiring all bits of the equivalent BAR to 0.

For a bridge function that is permitted to implement EA based on the rules above, it is permitted, but not required, for the bridge function to use EA mechanisms to indicate resource ranges that are located behind the bridge Function (see Section 6.9.1.2). In the example shown in in **Figure 6-1**, the bridge above Bus N is permitted to use EA mechanisms to indicate the resources used by the two functions in “Si component C”, or that bridge is permitted to not indicate the resources used by the two functions in “Si component C”. System firmware/software must comprehend that such bridge functions are not required to indicate inclusively all resources behind the bridge, and as a result system firmware/software must make a complete search of all functions behind the bridge to comprehend the resources used by those functions.

A Function with an Expansion ROM is permitted use the existing mechanism or the EA mechanism, but is not permitted to support both. If a Function uses the EA mechanism (EA entry with BEI of 8), the Expansion ROM Base Address Register (offset 30h) must be hardwired to 0. The Enable bit of the EA entry is equivalent to the Expansion ROM Enable bit. If a Function uses Expansion ROM Base Address Register mechanism , no EA entry with a BEI of 8 is permitted.

The requirements for enabling and/or disabling the decode of I/O and/or Memory ranges are unchanged by EA, including but not limited to the Memory Space and I/O Space enable bits in the Command register.

Any resource allocated using EA must not overlap with any other resource allocated using EA. except as permitted above for identifying permitted address ranges for resources behind a bridge.

6.9.1. Enhanced Allocation (EA) Capability Structure

Each function that supports the Enhanced Allocation mechanism must implement the Enhanced Allocation capability structure.

Each field is defined in the following sections. Reserved registers must return 0 when read and write operations must have no effect. Read-only registers return valid data when read, and write operations must have no effect.

6.9.1.1. Enhanced Allocation Capability First DW

The first DW of the Enhanced Allocation capability is illustrated in Figure 6-3, and is documented in the following table.

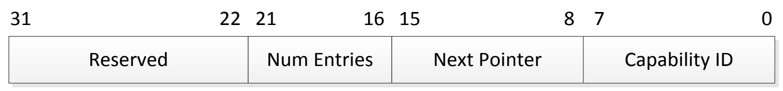


Figure 6-2: First DW of Enhanced Allocation Capability

Bits	Field	Description
7:0	CAP_ID	Must be set to 14h to indicate Enhanced Allocation capability. This field is read only.
15:8	NXT_PTR	Pointer to the next item in the capabilities list. Must be NULL for the final item in the list. This field is read only.
21:16	Num Entries	Number of entries following the first DW of the capability. Value of 00 0000b is permitted and means there are no entries. This field is read only.

6.9.1.2. *Enhanced Allocation Capability Second DW [Type 1 Functions Only]*

For Type 1 functions only, there is a second DW in the capability, preceding the first entry. This second DW must be included in the Enhanced Allocation Capability whenever this capability is implemented in a Type 1 Function. The second DW of the Enhanced Allocation capability is illustrated in Figure 6-3, and is documented in the following table.

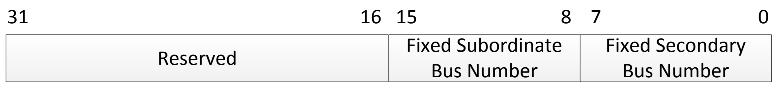


Figure 6-3: Second DW of Enhanced Allocation Capability

Bits	Field	Description
7:0	Fixed Secondary Bus Number	If at least one Function that uses EA is located behind this function, then this field must be set to indicate the bus number of the PCI bus segment to which the secondary interface of this Function is connected. If no Function that uses EA is located behind this function, then this field must be set to all 0's. This field is HwInit.
15:8	Fixed Subordinate Bus Number	If at least one Function that uses EA is located behind this function, then this field must be set to indicate the the bus number of the highest numbered PCI bus segment which is behind this Function. If no Function that uses-EA is located behind this function, then this field must be set to all 0's. This field is HwInit.

6.9.1.3. *Enhanced Allocation Per-Entry Format*

The first DW of each entry in the Enhanced Allocation capability is illustrated in **Figure 6-4**, and is defined in the following table.

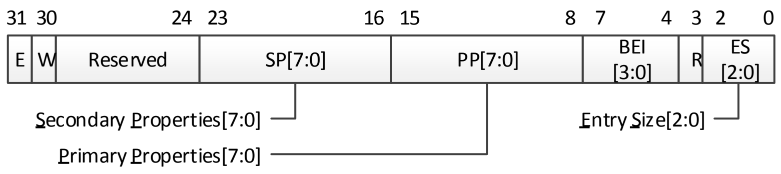


Figure 6-4: First DW of Each Entry for Enhanced Allocation Capability

Bits	**Field**	**Description**
2:0	Entry Size	Number of DW following the initial DW in this entry. When processing this capability, software is required to use the value in this field to determine the size of this entry, and if this entry is not the final entry, the start of the following entry in the capability. This requirement must be strictly followed by software, even if the indicated entry size does not correspond to any entry defined in this specification. Value of 000b indicates only the first DW (containing the Entry Size field) is included in the entry. This field is HwInit.
7:4	BEI	BAR Equivalent Indicator – This field indicates the equivalent BAR for this entry. Specific rules for use of this field are given in the text following this table. BEI Value — Description 0 — Entry is equivalent to BAR at location 10h 1 — Entry is equivalent to BAR at location 14h 2 — Entry is equivalent to BAR at location 18h 3 — Entry is equivalent to BAR at location 1Ch 4 — Entry is equivalent to BAR at location 20h 5 — Entry is equivalent to BAR at location 24h 6 — Permitted to be used by a Type 1 function only, optionally used to indicate a resource that is located behind the Function 7 — Equivalent Not Indicated 8 — Expansion ROM Base Address 9-14 — Entry relates to VF BARs 0-5 respectively 15 — Reserved – Software must treat values in this range as “Equivalent Not Indicated” This field is HwInit.
15:8	Primary Properties	Indicates the entry properties as defined in Table 6-1. This field is HwInit.
23:16	Secondary Properties	Optionally used to indicate a different but compatible entry property, using properties as defined in Table 6-1. This field is HwInit.
30	W	Writable – 1b indicates that the Base, MaxOffset and Field Size fields for this entry are RW, 0b indicates those fields are HwInit

31	E	Enable for this entry – 1b indicates enabled, 0b indicates disabled. If system software disables this entry, the resource indicated must still be associated with this function, and it is not permitted to reallocate this resource to any other entity. When system software writes or attempts to write to this bit, it must perform a read-modify-write such that all values read from all other bits of this DW are preserved. This field is permitted to be implemented as HwInit for functions that require the allocation of the associated resource, or as RW for functions that can allow system software to disable this resource, for example if BAR mechanisms are to be used instead of this resource.

Rules for use of BEI field:

- A Type 0 Function is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0-5, 7 or 8.
- A Physical Function (Type 0 Function that supports SR-IOV) is permitted to use EA to allocate resources for its associated Virtual Functions, and such resources must indicate a BEI value of 9-14.
- A Type 1 (bridge) function is permitted to use EA to allocate resources for itself, and such resources must indicate a BEI value of 0, 1 or 7.
- A Type 1 function is permitted but not required to indicate resources mapped behind that Function, but if such resources are indicated by the Type 1 function, the entry must indicate a BEI value of 6.
- For a 64-bit Base Address register, the BEI indicates the equivalent BAR location for lower DWORD.
- For Memory or I/O BARs where the Primary or Secondary Property is 00h, 01h or 02h, it is permitted to assign the same BEI in the range of 0-5 once for a range where Base + MaxOffset is below 4GB, and again for a range where Base + MaxOffset is greater than 4GB; It is not otherwise permitted to assign the same BEI in the range 0-5 for more than one entry.
- For Virtual Function BARs where the Primary or Secondary Property is 03h or 04h it is permitted to assign the same BEI in the range of 0-5 once for a range where Base + MaxOffset is below 4GB, and again for a range where Base + MaxOffset is greater than 4GB; It is not otherwise permitted to assign the same BEI in the range 0-5 for more than one VF entry.
- For all cases where two entries with the same BEI are permitted, Software must enable use of only one of the two ranges at a time for a given Function.
- It is permitted for an arbitrary number of entries to assign a BEI of 6 or 7.

- At most one entry is permitted with a BEI of 8; If such an entry is present, the Expansion ROM Base Address Register must be hardwired to 0.
- For Type 1 functions, BEI values 2 through 5 are reserved.

The **Figure 6-5** illustrates the format of a complete Enhanced Allocation entry for a Type 0 function. For the Base and MaxOffset fields, bit 1 indicates if the field is a 32b (0) or 64b (1) field.

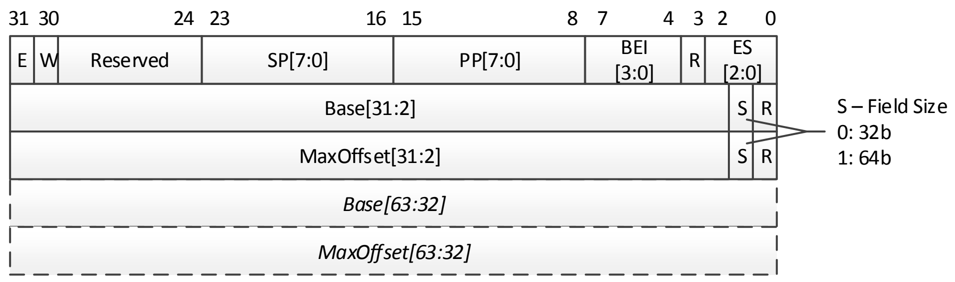


Figure 6-5: Format of Entry for Enhanced Allocation Capability

The value in the Base field ([63:2] or [31:2]) indicates the DW address of the start of the resource range. Bits [1:0] of the address are not included in the Base field, and must always be interpreted as 00b.

The value in the Base field plus the value in the MaxOffset field ([63:2] or [31:2]) indicates the address of the last included DW of the resource range. Bits [1:0] of the MaxOffset are not included in the MaxOffset field, and must always be interpreted as 11b.

For the Base and MaxOffset fields, when bits [63:32] are not provided then those bits must be interpreted as all 0's.

Although it is permitted for a Type 0 Function to indicate the use of a range that is not naturally aligned and/or not a power of two in size, some system software may fail if this is done. Particularly for ranges that are mapped to legacy BARs by indicating a BEI in the range of 0 to 5, it is strongly recommended that the Base and MaxOffset fields for a Type 0 Function indicate a naturally aligned region.

The Primary Properties[7:0] field must be set by hardware to identify the type of resource indicated by the entry. It is strongly recommended that hardware set the Secondary Properties[7:0] to indicate an alternate resource type which can be used by software when the Primary Properties[7:0] field value is not comprehended by that software, for example when older system software is used with new hardware that implements resources using a value for Primary Properties that was reserved at the time the older system software was implemented. When this is done, hardware must ensure that software operating using the resource according to the value indicated in the Secondary Properties field will operate in a functionally correct way, although it is not required that this operation will result in optimal system performance or behavior.

The Primary Properties[7:0] and Secondary Properties[7:0] fields are defined in the following table:

Table 6-1 Enhanced Allocation Entry Field Value Definitions for both the Primary Properties and Secondary Properties Fields

Value (h)	Resource Definition
00	Memory Space, Non-Prefetchable. The corresponding Base and MaxOffset fields are HwInit.
01	Memory Space, Prefetchable. The corresponding Base and MaxOffset fields are HwInit.
02	I/O Space. The corresponding Base and MaxOffset fields are HwInit
03	For use only by Physical Functions to indicate resources for Virtual Function use, Memory Space, Prefetchable. The corresponding Base and MaxOffset fields are HwInit.
04	For use only by Physical Functions to indicate resources for Virtual Function use, Memory Space, Non-Prefetchable. The corresponding Base and MaxOffset fields are HwInit.
05	For use only by Type 1 Functions to indicate Memory, Non-Prefetchable, for Allocation Behind that Bridge. The corresponding Base and MaxOffset fields are HwInit.
06	For use only by Type 1 Functions to indicate Memory, Prefetchable, for Allocation Behind that Bridge. The corresponding Base and MaxOffset fields are HwInit.
07	For use only by Type 1 Functions to indicate I/O Space for Allocation Behind that Bridge. The corresponding Base and MaxOffset fields are HwInit.
08-FE	Reserved for future use; System firmware/software must not write to this entry, and must not attempt to interpret this entry or to use this resource. When software reads a Primary Properties value that is within this range, is it strongly recommended that software treat this resource according to the value in the Secondary Properties field, if that field contains a non-reserved value.
FD	Memory Space Resource Unavailable For Use - – System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose. The corresponding Base and MaxOffset fields are HwInit.
FE	I/O Space Resource Unavailable For Use - – System firmware/software must not write to this entry, and must not attempt to use the resource described by this entry for any purpose The corresponding Base and MaxOffset fields are HwInit.
FF	Entry Unavailable For Use – System firmware/software must not write to this entry, and must not attempt to interpret this entry as indicating any resource. It is strongly recommended that hardware use this value in the Secondary Properties field to indicate that for proper operation, the hardware requires the use of the resource definition indicated in the Primary Properties field .

The following figures illustrate the layout of Enhanced Allocation entries for various cases.

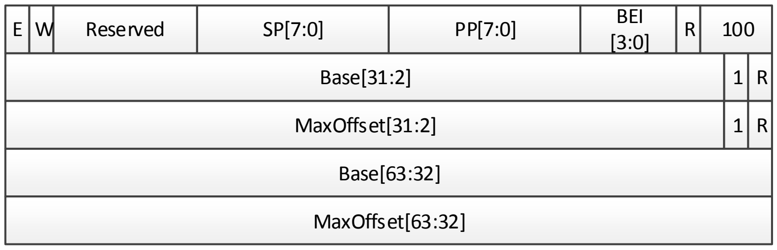


Figure 6-6: Example Entry with 64b Base and 64b MaxOffset

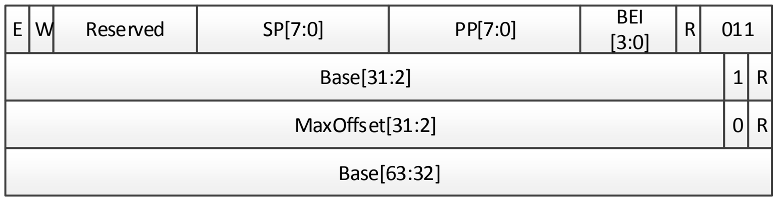


Figure 6-7: Example Entry with 64b Base and 32b MaxOffset

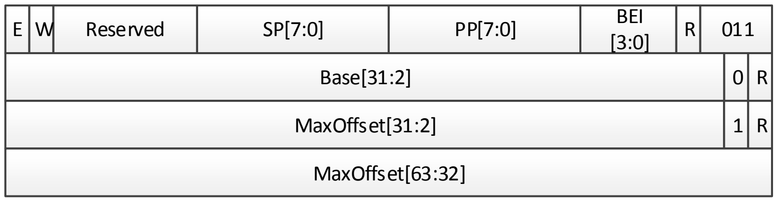


Figure 6-8: Example Entry with 32b Base and 64b MaxOffset

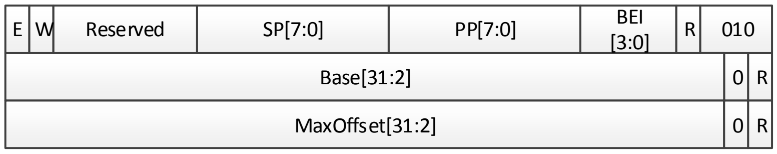


Figure 6-9: Example Entry with 32b Base and 32b MaxOffset

In Appendix G:

These addresses are not requested using a Base Address register or Enhanced Allocation capability but are assigned by initialization software. If a device identifies itself as a legacy function (class code), the initialization software grants the device permission to claim the I/O legacy addresses by setting the device's I/O Space enable bit.

In Appendix I:

… Location of the control/status registers is identified by providing the index (a value between 0 and 5) of the Base Address register (or, when using Enhanced Allocation capability, the corresponding BEI) that defines the address range that contains the registers …

For PCI Code and ID Assignment Specification, Modify Section 3 as shown:

Table 6-2: Capability IDs

ID	**Capability**
…	
13h	Advanced Features (AF) – Full documentation of this feature can be found in the *Advanced Capabilities for Conventional PCI ECN*.
14h	Enhanced Allocation
~~14~~15h-FFh	Reserved