**# Title:**

Describing hot-pluggable memory

**# Status:**

Draft

**# Document:**

ACPI Specification Version 6.next

**# License:**

SPDX-License-Identifier: CC-BY-4.0

**# Submitters:**

* Leo Duran (AMD)
* Samer El-Haj-Mahmoud (Arm)
* Thanu Rangarajan (Arm)
* Michael A Rothman (Intel)
* TianoCore Community (<https://www.tianocore.org>)

**# Summary of the change**

This ECR is related to hot-pluggable memory. There are some fundamental requirements for describing hot-pluggable memory, as follows:

* The UEFI memory map cannot be used to describe memory that is present at boot-time but hot-removable at runtime. That is because it is possible for the OS to allocate the memory for non-relocatable data during early kernel boot phase, such as page tables. It is also possible for early boot components such as boot loaders or Option ROMs to allocate that memory for data that the OS does not understand and cannot touch/relocate.
* The UEFI memory map cannot be used to describe memory that is not present or enabled at boot-time and can be either hot-added or brought online at runtime. This memory can thus not be used by early boot components.

In conclusion, the UEFI memory map must *only* be used to describe memory that is statically present and enabled at boot and thus available for use. Hot-pluggable memory must be described using ACPI. However, the ACPI specification does not explicitly call these points out today, which has led to ambiguity and various interpretations.

This ECR is proposing the following changes/enhancements:

* Clarifying language in the specification on the mutual exclusivity between the UEFI memory map and hot-pluggable memory.
* Mandating the use of the ACPI memory device object for describing hot-pluggable memory when the memory does not have a native hot-plug mechanism defined, where the Extended Address Space resource descriptor is used to specify the memory attributes and memory type of the hot-pluggable address range. A reference ASL example is also provided. An example of memory that has an associated native hot-plug mechanism is CXL [x].
* Mandating that the memory device object must carry an \_STA method as well as either or both of \_EJ0 and \_DIS methods to indicate to the OS that the range it is describing is hot-pluggable.
* Enhancing the Extended Address Space descriptor macro to permit use of resource types AddressRangeMemory through AddressRangeNVS (values 0x00 through 0x03).

**# Benefits of the change**

Improved description of system memory to the OS, and more informed use of that memory by the OS and its sub-components.

**# Impact of the change**

Backward compatibility with existing OS implementations is ensured, however changes will be required for the OS to comprehend the proposed mechanisms in this ECR. Firmware must follow the principles outlined in this ECR.

**# Detailed description of the change [normative updates]**

Existing text

New text

~~Deleted Text~~

Text to be discussed

**5.2.16.2 Memory Affinity Structure**

…

**Table 5.59: Flags - Memory Affinity Structure**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Bit Length** | **Bit Offset** | **Description** |
| … | … | … | … |
| Hot Pluggable | 1 | 1 | The information conveyed by this bit depends on the value of  the Enabled bit. If the Enabled bit is set and the Hot Pluggable  bit is also set. The system hardware supports hot-add and hot-remove of this memory region. If the Enabled bit is set and  the Hot Pluggable bit is clear, the system hardware does not  support hot-add or hot-remove of this memory region. If the  Enabled bit is clear, the OSPM will ignore the contents of the  Memory Affinity Structure.  If this bit is set and there is no native mechanism for hot-plug of the memory ranges described by this structure, there must exist a memory device (see Section 9.11), where the following conditions must be satisfied:   * This memory region must be a part of the memory range described by the memory device. * The memory device must satisfy the hot-pluggability conditions outlined in Section 9.x.1.   Please see Section 9.x for examples of memory that has associated native hot-plug mechanisms. |
| … | … | … | … |

**9.x Memory Devices**

…

Memory devices allow a platform to convey dynamic properties of memory to OSPM and are required when a platform supports the addition or removal of memory while the system is active or when the platform supports memory bandwidth monitoring and reporting (see Section 9.11.2). Memory devices are assigned a PNPID of PNP0C80.

For the active memory additional and removal use-case, the memory device object is only required if there is no other native mechanism for performing the hot-add or hot-remove operations. For example, hot-plug of CXL-attached memory employs CXL-defined mechanisms and, as such, a memory device object is not required for such memory.

…

**9.x.1 Hot-plug Indication**

If the memory device is created for the purpose of describing hot-pluggable memory, it must always carry the \_STA, as well as either \_EJ0 or \_DIS methods, or both. OS can use the presence of these methods as an indication that the memory range is hot-pluggable. In addition, there must be a matching memory affinity structure in the SRAT table that has the Hot-pluggable flag set. See Section 5.2.16.2 for further details on this flag. The expression for confirming hot-pluggable property is as follows:

Is Hot-pluggable = \_STA && (\_EJ0 || \_DIS);

**9.x.~~1~~3 Address Decoding**

…

Memory devices must provide a \_CRS object that describes the physical address space that the memory decodes. If

the memory can decode alternative ranges in physical address space, the devices may also provide \_PRS, \_SRS and

\_DIS objects. Other device objects may also apply if the device can be ejected.

The physical address space described by \_CRS object must be described using the Extended Address Space Resource Descriptor macro. The *TypeSpecificAttributes* (\_ATT) field of the descriptor might then be used to set the EFI memory attributes that apply to the memory. In the case of memory hot-add, the OS can then use the \_ATT field information to understand how the memory must be used after it has been added. This enables hot-plug support for specific-purpose memory (SPM) and persistent memory. Since the \_ATT field is optional, the OS must consider its absence to mean that the memory is by default cacheable memory with EFI attributes set to EFI\_MEMORY\_WB.

The default UEFI memory type for memory described by memory devices is *AddressRangeMemory*. Please see the UEFI specification for more information on this memory type.

Hot-pluggable persistent memory ranges must not be described using this mechanism. They should instead be described using the NFIT table and related methods specific to persistent memory.

**9.x.y Hot-pluggable Memory Description Illustrated**

The following is an example that shows a hot-pluggable memory module that is mapped at offset 0x10000000, and can decode up to 0x20000000 bytes of memory. The memory module has its *TypeSpecificAttributes* field set to EFI\_MEMORY\_SP, to indicate to the OS that it is meant for specific-purpose usage.

|  |
| --- |
| Scope (\\_SB){  Device (MEM0) {  Name (\_HID, EISAID ("PNP0C80"))  Method (\_STA) {Return (ST01)} // Status stored in local  Variable called ST01  Method (\_EJ0) {}  Method (\_OST) {}  Name (\_CRS, ResourceTemplate () {  ExtendedSpace (  0x00, // 0x00 = Normal Memory Range  ResourceConsumer,  Bits[4:3] = 00b, // AddressRangeMemory  MinFixed,  MaxFixed,  Cacheable,  0xFFFFFFF,  0x10000000,  0x30000000,  0,  0x20000000,  EFI\_MEMORY\_SP,// Specific-purpose memory  )  } )  }  } |

**CHAPTER FIFTEEN**

**SYSTEM ADDRESS MAP INTERFACES**

…

Lastly, if memory resources may be added or removed dynamically, memory devices are defined in the ACPI Namespace conveying the resource information described by the memory device (see Memory Devices). Such memory resources must not be described using UEFI memory descriptors. See section 15.4 for more details.

**15.4 UEFI Assumptions and Limitations**

…

* Hot-pluggable memory, or memory that can be dynamically added or removed at runtime, must not be described using EFI memory descriptors. The EFI memory descriptors are the only source of information regarding system address map for early boot components such as boot loaders, Option ROMs and early OS kernel code. These components might inadvertently attempt allocations from that memory for storing non-data structures during OS initialization. If the memory is then removed, it can cause OS failure if the data structures are non-relocatable. Conversely, if the memory is not present or enabled at the time of OS boot and intended to be added only later at runtime, any allocations from it can result in a catastrophic failure. For hot-pluggable memory, therefore, it is mandatory for the firmware to only describe the memory using the ACPI memory device object.

**6.4 Resource Data Types for ACPI**

…

**6.4.3. Large Resource Data Type**

…

**6.4.3.5.4 Extended Address Space Descriptor**

**Type 1, Large Item Value 0xB**

…

|  |  |  |
| --- | --- | --- |
| **Offset** | **Field Name** | **Definition** |
| … | … | … |
| Byte 5 | Type Specific Flags | Optional f~~F~~lags that are specific to each resource type. The meaning of the flags in  this field depends on the value of the Resource Type field (see above). For  the Memory Resource Type, the definition is defined in Resource Type  Specific Flags. For other Resource Types, refer to the existing definitions  for the Address Space Descriptors. |
| … | … | … |
| Byte 48 | Type Specific Attribute,  \_ATT bits[7:0] | Optional Aattributes that are specific to each resource type. The meaning of the attributes in this field depends on the value of the Resource Type field  (see above). For the Memory Resource Type definition, see Type Specific  Attributes. For other Resource Types, this field is reserved to 0. |
| … | … | … |

**19.6.44 ExtendedSpace (Extended Address Space Resource Descriptor Macro)**

**…**

*ResourceType* evaluates to an 8-bit integer that specifies the type of this resource. Acceptable values are 0x00 through 0x03 and 0xC0 through 0xFF.

*TypeSpecificFlags* evaluates to an 8-bit integer. The flags are specific to the ResourceType. This field is optional. If absent, it represents memory space of type AddressRangeMemory.